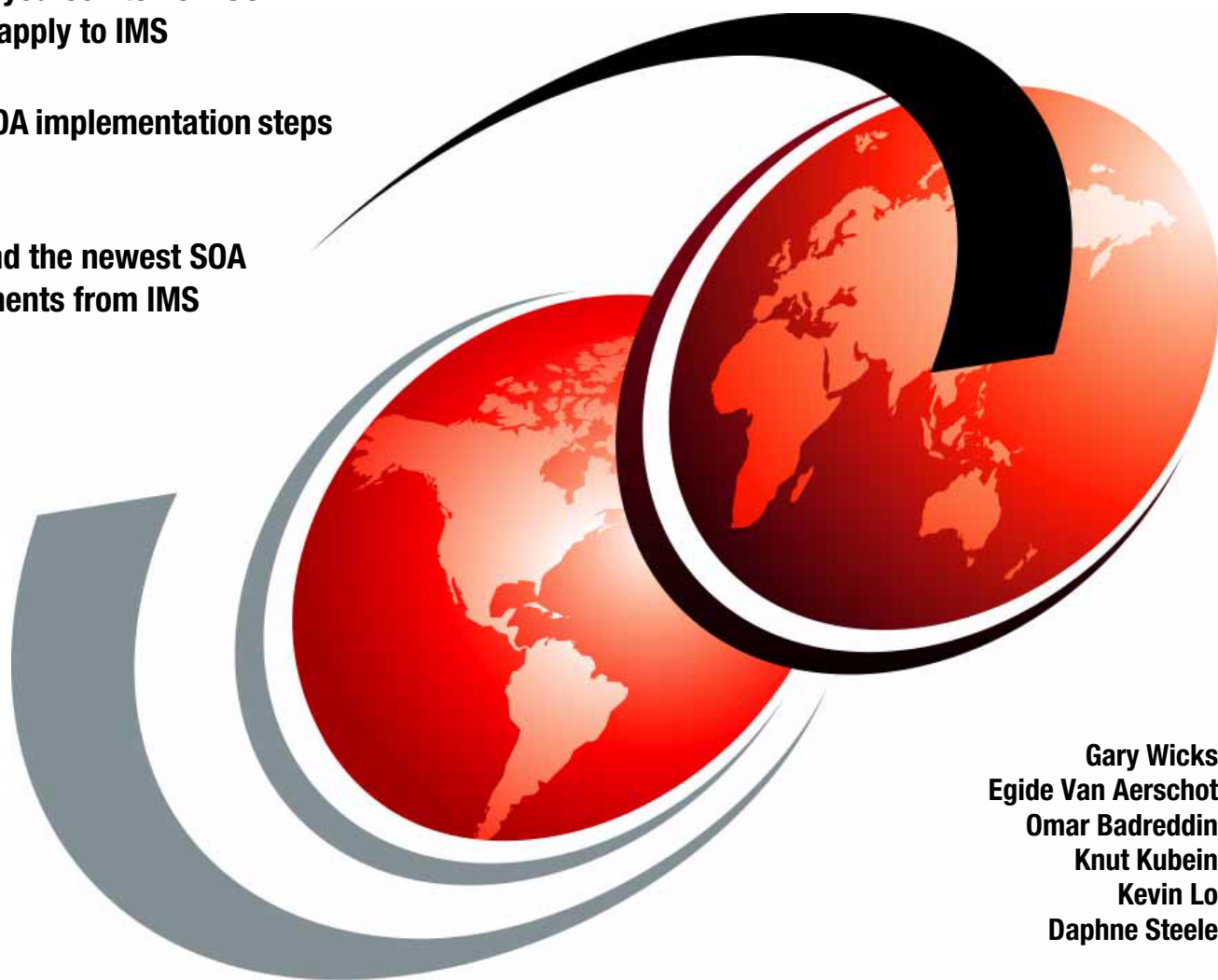


Powering SOA Solutions with IMS

Introduce yourself to how SOA
concepts apply to IMS

Identify SOA implementation steps

Understand the newest SOA
enhancements from IMS



Gary Wicks
Egide Van Aerschot
Omar Badreddin
Knut Kubein
Kevin Lo
Daphne Steele

Redbooks



International Technical Support Organization

Powering SOA Solutions with IMS

March 2009

Note: Before using this information and the product it supports, read the information in “Notices” on page xi.

First Edition (March 2009)

This edition applies to Version 10 (program number 5635-A01) and Version 11 Quality Partnership Program (QPP) level (program number 5635-A02) of IBM Information Management System.

© Copyright International Business Machines Corporation 2009. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	xi
Trademarks	xii
Preface	xiii
The team that wrote this book	xiii
Become a published author	xv
Comments welcome	xv
Part 1. SOA and IMS: A powerful business combination	1
Chapter 1. SOA and IMS: The big picture	3
1.1 What is service-oriented architecture	4
1.1.1 Major components of SOA	4
1.1.2 Major roles and activities in a SOA	5
1.1.3 SOA and standards	6
1.2 Terminology	7
1.3 The value of including existing IMS assets into SOA	11
1.3.1 IMS Connect and IMS Connect Extensions	15
1.3.2 The IMS SOA Integration Suite	16
1.3.3 IMS SOAP Gateway	16
1.3.4 IMS TM Resource Adapter	18
1.3.5 IMS MFS Web Solutions: Web Enablement, services, and SOA support	19
1.3.6 IMS Web 2.0 Solution: IMS Info 2.0 for InfoSphere MashupHub	21
1.3.7 The DLIModel utility	23
1.4 IMS Open Database	24
1.5 DataPower and IMS	26
1.5.1 The value of using DataPower	26
Chapter 2. Development phases of IMS SOA projects	27
2.1 IMS as a SOA integration focal point	28
2.2 The value of IMS service reuse	30
2.3 Styles of business transformations	32
2.4 Methodology of the development of SOA projects	32
2.4.1 The Rational Unified Process	33
2.4.2 Process modeling for SOA through SOMA	34
2.5 Phases of IMS SOA projects	35
2.5.1 Phase 1: Determining if a SOA project is worth considering	35
2.5.2 Phase 2: Developing the SOA project office and plan	36
2.5.3 Phase 3: Integrating the IMS modernization project into a governance model	38
2.5.4 Phase 4: Identifying candidate services and flows	40
2.5.5 Phase 5: Identify your specific IMS assets	44
2.5.6 Phase 6: Evaluating architectural changes and technologies to utilize	48
2.5.7 Phase 7: Creating an Enterprise Service Bus	49
2.5.8 Phase 8: Selecting a pilot project	50
2.5.9 Phase 9: Implementing, testing, and monitoring IT services	51
Chapter 3. SOA Tooling	57
3.1 Overview of tools support	58
3.2 IMS SOA core tools	58

3.2.1 Rational Application Developer	58
3.2.2 Rational Developer for System z	58
3.2.3 Understanding the Rational Developer for System z workbench environment.	59
3.2.4 WebSphere Application Server	61
3.3 IMS SOA supporting tools	63
3.3.1 SOA-enabled Business Process Management	63
3.3.2 WebSphere Business Modeler	65
3.3.3 WebSphere Service Registry and Repository	65
3.3.4 WebSphere Integration Developer	65
Part 2. Creating, deploying, and managing IMS services	69
Chapter 4. Example of deploying an IMS Service	71
4.1 Background to SOAP Gateway	72
4.1.1 SOAP Gateway development and deployment	72
4.2 Installing IMS SOAP Gateway on z/OS	73
4.2.1 Installing and setting up	73
4.2.2 Preparing for installation	74
4.2.3 Setting up	76
4.2.4 Installing the SOAP Gateway	78
4.2.5 Post installation steps	81
4.3 Setting up IMS Connect and IMS OTMA	83
4.4 Accessing an IMS SOAP Gateway sample application	84
Chapter 5. Securing IMS services	85
5.1 Security ISO and standards	86
5.2 Layers of SOA security	86
5.3 Security management	87
5.3.1 Component-managed EIS sign-on	87
5.3.2 Container-managed EIS sign-on	88
5.3.3 Connection factory custom properties	89
5.4 The WebSphere security mechanism	89
5.4.1 Web container security	90
5.5 The value of Tivoli Access Manager WebSEAL	91
5.6 Open Transaction Manager Access security considerations	92
5.6.1 Resume transaction pipe security	93
5.6.2 Transaction member level security	93
5.6.3 Accessor environment element Aging Value	94
5.6.4 IMS OTMA callout security	94
5.7 IMS Connect security considerations	95
5.7.1 IMS Connect SSL connections	97
5.8 IMS TM Resource Adapter security considerations	98
5.8.1 IMS connection factory properties	98
5.9 IMS SOAP Gateway security considerations	100
5.9.1 SOAP Gateway connection bundle	100
5.10 Configuring the SOAP Gateway for SSL	101
Chapter 6. Problem management for IMS services	105
6.1 Life cycle solutions to ensure functional quality	106
6.2 IBM IT life cycle management tools	106
6.2.1 Rational ClearQuest	107
6.2.2 Rational Functional Tester	108
6.3 Omegamon XE and IMS problem management	108
6.4 Problem determination tooling	109

6.4.1	The IBM problem determination tools for System z	109
6.5	IMS Connect tracing	111
6.6	IMS TM Resource Adapter logging and tracing	112
6.7	SOAP Gateway tracing	113
6.8	TCP/IP problem analysis	114
6.9	Logging and tracing on WebSphere Application Server	114
Chapter 7. Performance considerations for IMS Services		117
7.1	Measuring and monitoring performance	118
7.1.1	Best general practices for measuring performance	118
7.2	Performance Measurement Infrastructure	118
7.2.1	The performance review process	119
7.3	Performance tuning for TCP/IP	121
7.3.1	Important client-side SETSOCKOPT TCP/IP parameters	121
7.3.2	PROFILE.TCPIP specifications	121
7.3.3	BPXPRMxx member in SYS1.PARMLIB	122
7.3.4	XCF tuning	122
7.3.5	The Domain Name System configuration	122
7.3.6	Send and receive buffer specifications	122
7.3.7	Hipersockets	123
7.4	Performance tuning for the WebSphere Application Server	124
7.4.1	Queuing network tuning	124
7.4.2	WebSphere plug-in performance	124
7.4.3	Using the dynamic cache service to improve performance	127
7.4.4	Workload Manager tuning for WebSphere Application Server	127
7.4.5	Performance considerations for WebSphere connectors	127
7.5	IMS Connect performance parameters	128
7.6	IMS Connect Extensions	129
7.7	Language Environment performance discussion	129
7.8	Performance monitoring tools	130
7.8.1	Resource Measurement Facility	131
7.8.2	Eclipse Test and Performance Tools Platform	132
7.8.3	Tivoli Performance Viewer	133
7.8.4	Tivoli Composite Application Monitor	133
7.8.5	WebSphere Business Monitor	134
7.8.6	Tivoli Omegamon XE for IMS Connect	138
7.8.7	Rational Performance Tester for z/OS	138
7.8.8	JMeter	138
7.8.9	IMS Performance Analyzer	139
7.8.10	IMS Problem Investigator	140
Part 3. SOA enhancements in IMS and the IMS SOA Integration Suite		143
Chapter 8. IMS SOA enhancements		145
8.1	On-demand and SOA-related IMS Version 10 features	146
8.2	Post IMS Version 10 GA features introduced through maintenance	147
8.3	Features introduced in IMS Version 11	148
8.4	Introduction to IMS Callout support	151
8.4.1	Using the DL/I ISRT ALTPCB call	151
8.4.2	Using CPI-C/APPC calls	152
8.4.3	Use of TCP/IP sockets calls	153
8.4.4	Using WebSphere MQ calls	154
8.4.5	Using TM Resource Adapter, SOAP Gateway, and RYO callout support	155
8.4.6	Highlights of IMS synchronous callout support	155

8.4.7	Stages to implement the synchronous call function	157
8.4.8	Creating or modifying your IMS application to issue an ICAL call.	158
8.4.9	Defining the OTMA descriptor for destination routing	158
8.4.10	Software requirements for synchronous callout support.	159
8.4.11	Synchronous callout restrictions	160
8.5	Open Transaction Manager Access overview.	160
8.5.1	Activating Open Transaction Manager Access	161
8.5.2	Open Transaction Manager Access transaction pipe	161
8.5.3	Open Transaction Manager Access enhancements in Version 10	162
8.5.4	Open Transaction Manager Access enhancements in Version 11	163
8.5.5	Open Transaction Manager Access descriptors	164
Chapter 9. IMS Connect and IMS Connect Extensions		167
9.1	IMS Connect and SOA	168
9.2	TCP/IP.	169
9.2.1	Commit and Synclevel	170
9.2.2	Persistent Sockets	170
9.3	High-level overview of IMS Connect	171
9.4	IMS Connect configurations	172
9.5	What is new in IMS Connect Version 10.	175
9.6	What is new in IMS Connect Version 11.	176
9.6.1	Integrated IMS Connect enhancements for IMS DB.	176
9.6.2	Integrated IMS Connect enhancements for IMS TM.	177
9.7	IMS as a service provider	179
9.7.1	IMS Control Center	180
9.7.2	Roll Your Own client program	181
9.7.3	IMS TM Resource Adapter client program	183
9.7.4	Soap Gateway	185
9.8	IMS as a service consumer.	185
9.8.1	IMS TM Resource Adapter asynchronous callout mode.	186
9.8.2	The IMS TM Resource Adapter synchronous callout mode	186
9.8.3	IMS SOAP Gateway callouts	187
9.9	IMS Connect Extensions.	188
9.9.1	IMS Connect Extensions components	189
9.9.2	Integrating IMS Connect Extensions with IMS Connect	193
9.9.3	Using IMS Connect Extensions with other IMS tooling.	196
9.10	IMS Connect as a database router	197
9.11	IMS Connect use in a Sysplex and IMSplex.	199
9.11.1	IMS Connect support for z/OS Sysplex Distributor	199
9.11.2	IMS Connect support for IMSplex and the IMS Control Center	200
9.11.3	IMS Connect support for IMSplex and shared queues	201
9.12	Conclusion	201
Chapter 10. The IMS SOAP Gateway		203
10.1	IMS SOAP Gateway implementation overview.	204
10.2	IMS SOAP Gateway and your IMS applications	207
10.2.1	XML-to-bytes and bytes-to-XML	208
10.3	What is new in IMS SOAP Gateway	208
10.4	Asynchronous Callout with SOAP Gateway	209
10.4.1	Invoking the Web Service operation	209
10.4.2	Returning the callout response message to IMS	210
10.4.3	Web Services callout scenarios for Asynchronous Callout.	210
10.5	Synchronous Callout with IMS SOAP Gateway	216

10.5.1	Detailing Synchronous Callout with IMS SOAP Gateway	217
10.5.2	Synchronous Callout User Scenarios	219
10.6	Multi-segment support	222
10.7	Security enhancements	222
10.8	IMS SOAP Gateway features and compatibilities	223
Chapter 11.	The IMS TM Resource Adapter	225
11.1	Key features of the IMS TM Resource Adapter	226
11.1.1	Features introduced in Version 10.2	227
11.1.2	Features introduced in Version 11	227
11.2	Installing the IMS TM Resource Adapter	229
11.3	Call-in request support	229
11.4	Callout request support	233
11.4.1	Destinations for callouts through TM Resource Adapter	234
11.4.2	Asynchronous callout requests	235
11.4.3	Synchronous callout requests	239
Chapter 12.	MFS Web solutions	245
12.1	Technologies behind MFS Web Solutions	246
12.1.1	Why MFS Web Solutions were developed	247
12.1.2	Sample Client experience with IMS Web Solutions	248
12.2	MFS Web Enablement	249
12.2.1	Prerequisites for MFS Web Enablement	250
12.2.2	How does MFS Web Enablement work	250
12.2.3	The MFS Adapter component of MFS Web Enablement	252
12.2.4	MFS Servlet and stylesheets	255
12.2.5	The MFS XML utility	256
12.2.6	Running MFS Web Enablement	257
12.2.7	Connection Management	260
12.2.8	Supported core MFS features	261
12.3	MFS SOA support	262
12.3.1	Why was MFS SOA Support developed	262
12.3.2	Development and runtime environments	263
12.3.3	Functional components of MFS SOA solutions	264
12.3.4	Creating Java EE resources and embedding the J2C bean	265
12.3.5	Conversational support	267
Chapter 13.	IMS Web 2.0 Solution	271
13.1	What is Web 2.0	272
13.1.1	Web 2.0 and IMS	272
13.1.2	Business benefits	273
13.1.3	Web 2.0 and services-oriented architecture	276
13.2	Definition of key terminology	278
13.2.1	What is a REST service	278
13.2.2	What is a feed	279
13.2.3	What is a widget	280
13.2.4	What is a mashup	280
13.3	IBM Mashup Center Enterprise Edition Version 1.1	281
13.3.1	InfoSphere MashupHub	282
13.3.2	Lotus Mashup	282
13.3.3	WebSphere sMash	283
13.4	IMS Web 2.0 Solutions architecture	283
13.4.1	Software requirements and restrictions	285
13.5	Creation of IMS RESTful service from an IMS application	286

13.5.1	Identifying the application design and requirements for the feed	286
13.5.2	Generating Metafiles (XML converter driver and correlator file)	286
13.5.3	Deploying the XML converter driver to IMS Connect	289
13.5.4	Creating a feed from an IMS application in InfoSphere MashupHub	291
13.6	IMS Web 2.0 and IMS On-demand	292
Chapter 14.	DLIModel Utility, DB Web Services, and XQuery	293
14.1	IMS databases in the SOA environment	294
14.2	Configuring for Java applications accessing IMS DB	297
14.2.1	DB Resource Adapters (IMS Version 11 and IMS Version 10 SPE)	297
14.2.2	Accessing DLI data on z/OS	298
14.2.3	Accessing IMS data from distributed environments	300
14.3	DLIModel Utility	301
14.3.1	Using the DLIModel Utility plug-in	303
14.4	Web Services for DLI databases (IMS V11 and IMS V10 with SPE)	306
14.5	IMS, XML, and XQuery	311
14.5.1	XML storage in IMS databases	312
14.5.2	SQL extensions for XML storage and retrieval	315
14.5.3	XQuery	316
14.6	Conclusion	319
Chapter 15.	IMS Open Database and Universal Drivers	321
15.1	IMS Open Database	322
15.1.1	Review of ODBA use	323
15.1.2	Open Database Manager	323
15.2	IMS Database Resource Adapters and the IMS Universal Drivers	325
15.3	Overview of how the IMS Universal drivers work with IMS	328
15.3.1	IMS Universal Drivers programming model	330
15.3.2	Details on IMS Universal Drivers	331
Part 4.	DataPower solutions for IMS	339
Chapter 16.	Using DataPower with IMS	341
16.1	IBM WebSphere DataPower XML Appliances	342
16.1.1	WebSphere DataPower XML Accelerator XA35	342
16.1.2	DataPower XML Security Gateway XS40	343
16.1.3	DataPower Integration Appliance XI50	343
16.2	DataPower deployment scenarios and use cases	344
16.2.1	Protocol and format bridging	346
16.2.2	Configuring and using DataPower	348
Part 5.	Appendixes	353
Appendix A.	Sample code snippets	355
A.1	Asynchronous callout to a StateLess Session Bean	355
A.2	Asynchronous callout to a Message Driven Bean	357
A.3	Synchronous callout to a StateLess Session Bean	360
A.4	Synchronous callout to a Message Driven Bean	361
A.5	Feed from an IMS application in MashupHub	362
A.6	WSDL files for a DLIModel generated Web Services Service	364
A.7	XSD files for a DLIModel generated Web Services Service	366
	Abbreviations and acronyms	371
	Related publications	373

IBM Redbooks	373
Other publications	373
Online resources	374
How to get Redbooks	377
Help from IBM	377
Index	379

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	FileNet®	Redbooks®
Ascential®	HiperSockets™	Redbooks (logo)  ®
CICS®	IBM®	RequisitePro®
ClearQuest®	InfoSphere™	ScriptAssure®
DataPower®	Language Environment®	System z®
DataStage®	Lotus®	Tivoli®
DB2®	OMEGAMON®	VTAM®
Distributed Relational Database Architecture™	OS/390®	WebSphere®
DRDA®	Parallel Sysplex®	z/OS®
Enterprise Workload Manager™	RACF®	z/VM®
	Rational®	zSeries®

The following terms are trademarks of other companies:

FileNet, and the FileNet logo are registered trademarks of FileNet Corporation in the United States, other countries or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

EJB, Enterprise JavaBeans, J2EE, Java, JavaBeans, JavaServer, JDBC, JDK, JNI, JRE, JSP, JVM, RSM, Solaris, Streamline, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Excel, Internet Explorer, Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

New application development tools and the IBM® service-oriented architecture capabilities for IMS can help your business improve the speed and agility of its development efforts. Both IMS and the IMS SOA Integration Suite support your on demand systems and your distributed IMS application environment.

Powering SOA Solutions with IMS provides background and explanations to clarify the choices and methodologies that are available to modernize your IMS applications and provide access to IMS data stores through non-traditional callers.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose, California, USA.

Gary Wicks is an IBM Certified I/T Specialist. He has 34 years of experience with IBM in software service and is currently on assignment as the Team Lead for the IMS development technical specialist group. He has a degree in Mathematics and Physics from the University of Toronto. His areas of expertise include IMS enablement in Parallel Sysplex® environments and he wrote several IBM Redbooks® publications over the last twelve years. His role for this book is as Team Leader.

Egide Van Aerschot has an Engineering degree in Electricity and Nuclear Physics from the University of Leuven, Belgium. He joined IBM in 1967 and was responsible for many computer installations related to tele-processing and database management in Belgium. In 1997, he moved from IBM Belgium to IBM France, where he works as an Architect and Consultant at the IBM Program Support Center in Montpellier. Since 1997, he specialized in Java™ and WebSphere® applications, mainly on z/OS® systems, and participated in many projects that are related to the Internet. Egide is co-owner of the patent "Methods, systems, program product for transferring program code between computer processes". Currently Egide is a contractor for the Zinteg Corporation.

Omar Badreddin is a researcher at the IBM Center for Advanced Studies and Research in Ottawa and a member of Complexity Reduction in Software Engineering research group (CRuiSE). Omar has five years of work experience with IBM as a Software Engineer and has delivered a number of ITSO workshops. His area of expertise includes virtualization, modeling, BPM, service-oriented architecture (SOA), and software globalization.

Knut Kubein is a Senior Advisory I/T Specialist with IBM Global Services in Germany. He has 36 years of experience with IBM large systems products working as a Technical Support Specialist, with 26 of those years devoted to IMS. He leads the IMS Support Team in EMEA. His areas of expertise include IMS data sharing, shared queues, and the Parallel Sysplex architecture. He supports large banking clients and other industrial IMS users. He co-authored several IBM Redbooks publications on parallel sysplex and performance with IMS.

Kevin Lo has a Master of Science degree in Information Networking from Carnegie Mellon University. As a Staff Software Engineer at the IBM Silicon Valley Laboratory in San Jose, California, he has five years of experience in the IMS SOA on-demand field and is currently a Development Lead of IMS Message Format Service (MFS) Web Solutions, which includes MFS SOA Support and MFS Web Enablement.

Daphne Steele has a computer science degree from Tennessee State University. She is a Staff Software Engineer at the IBM Silicon Valley Laboratory in San Jose, California. She has five years of experience at IBM, and is the Function Test Team Lead of the IMS SOA Integration Suite. Daphne has expertise in test automation using test automation tools, such as Rational® Functional Tester. In addition, she was instrumental in redefining the function and system test process for the IMS test organization by unifying its test plan and process.

Thanks to the following people for their contributions to this project:

Paolo Bruni
Rich Conway
Bob Haimowitz
Emma Jacobs
Sangam Racherla
International Technical Support Organization

Kyle Charlet
Nathan D Church
Himakar Chennapragada
Demetrios Dimatos
Kevin Flanigan
Haley Fung
Elvis Halcombie,
Kevin Hite
Chris Holtzi
Jenny Hung
Yee-Rong Lai
William Li
Barbara Klein
Rose Levin
Evgueni Liakhovitch
Chao-Lin Liu
Hiram K. Neal
Marilene A. Noronha
Danny Nguyen
Malaika Paquiot-Mose
Maria G. Querales
Richard Tran
Kevin Washington
IBM Silicon Valley Laboratory, San Jose, U.S.A

John T Gates Jr
Americas Software Sales IBM

Hélène Lyon
IMS and z/Middleware Technical Sales, IBM France NorthWest Africa & South Europe

Jacqueline Ryan
Portfolio Management Strategy, Leveling Information Marketing Management IBM

Norbert Bieberstein
IBM Sales & Distribution, Software Sales, Germany

We offer special thanks for Suzie Wendler and Ken Blackman of the IBM IMS Advanced Technical Support (ATS), Americas for some valuable material that was sourced for this book.

Become a published author

Join us for a two to six week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400



Part 1

SOA and IMS: A powerful business combination

IMS is the IBM high-performance application and data server for System z®. As it has been for the last 40 years, no other solution today offers the combination of extreme performance, scalability, rock-solid reliability, and runtime efficiency. When it comes to running core applications that are at the heart of business processing, most large corporations worldwide continue to depend on IMS.

Service-oriented architecture (SOA) defines the use of loosely coupled software services to support business processes. In 2005, the Gartner Group said that “in 2008, SOA will provide the basis for 80 percent of new IT transformation projects”¹. Whether this projection proves completely accurate, it is obvious that services that represent repeatable tasks using well-defined, standards-based interfaces are being developed in the majority of IT modernization projects.

In response to this paradigm shift in the IT industry, IMS enterprise On Demand SOA solutions were created to enable IMS as requester and responder to participate in SOA environments as an integration focal point.

In part 1, we discuss:

- ▶ Chapter 1, “SOA and IMS: The big picture” on page 3
- ▶ Chapter 2, “Development phases of IMS SOA projects” on page 27
- ▶ Chapter 3, “SOA Tooling” on page 57

¹ *Gartner's Positions on the Five Hottest IT Topics and Trends in 2005.*
Web site: http://www.gartner.com/DisplayDocument?doc_cd=125868

It is important to understand that we target this book to people who are relatively new to SOA and want to understand how the 'pieces' that make up this, at times overwhelming topic, fit together.

We want to introduce you to the newest facilities from IMS and its integration suite of products at the IMS Version 10 level and beyond. We do not explain the entire IMS SOA story because there are two excellent IBM Redbooks publications that you can refer to for details:

- ▶ *IMS Connectivity in an On Demand Environment: A Practical Guide to IMS Connectivity*, SG24-6794

This book is written to the IMS Version 9 level.

- ▶ *IBM IMS Version10 Implementation Guide A Technical Overview*, SG24-7526

This book provides the enhancements that IMS Version 10 introduced.

There were many Special Product Enhancements (SPEs) introduced into the product through the maintenance stream that augmented IMS SOA solutions. We discuss them in this publication.

Also, even though at the time of publication of this book, IMS Version 11 was not generally available, it was announced in IBM letter 208-258 and as such, we discuss the line items contained in Version 11 for your educational and planning purposes.



SOA and IMS: The big picture

IMS has been exceptionally successful over the last 40 years as a high availability and performing solution for core business transaction execution and database access. SOA presents opportunities for enhanced application development and open access to IMS applications and data stores. But this is a relatively new venture; and as such, we want to present the following topics in this chapter:

- ▶ Provide an overview of IMS SOA solutions and concepts
- ▶ Define important terminology used with SOA
- ▶ Understand how IMS can contribute and integrate in a SOA environment
- ▶ Introduce the key IMS product and IMS SOA Integration Suite middleware functions and tools that facilitate SOA solutioning

1.1 What is service-oriented architecture

Service-oriented architecture (SOA) is an architecture style that is centered around components, or services, with standardized interfaces. It is a methodology of designing and running the software portion of an information technology infrastructure so that it supports the various individual and interrelated functions that are needed to operate a particular enterprise.

SOA principles are not new and they are not an IT invention. To elaborate on a major principle that is associated with SOA, we provide the example of your DVD player. The DVD player can connect virtually to any TV and accepts almost any alkaline batteries of the appropriate size. In addition, the DVD player can play any DVD disk. This inter operability of functionality is achievable only when there are widely defined and accepted standardized component and interface designs. All of the DVD manufacturers know exactly how to manufacture a compliant DVD.

The DVD player example can easily be extended to the car manufacturing industry. Componentization and defined interfacing allows car manufacturers to out source the manufacturing of subcomponents of cars with a great deal of confidence that these components fit in their manufactured cars.

SOA helps bridge the business/IT gap and helps systems remain scalable and flexible while your business is growing and changing. SOA is focused on business processes, and although many legitimate approaches exist for software architecture, SOA is intended explicitly for business applications:

- ▶ SOA is about reuse
 - Not “out with the old, in with the new”
- ▶ SOA is about packaging
 - Taking what exists and structuring it differently such that future expansion is fast, easy, and cheap
- ▶ SOA is about liberating business from the constraints of technology
 - This provides both integration and redundancy of use opportunities
- ▶ SOA is about services
 - As a set of loosely coupled black-box components

1.1.1 Major components of SOA

SOA is more cohesive in design than traditional application-development solutions. Figure 1-1 on page 5 illustrates four main components of the enterprise architecture within the SOA framework. We look into these architecture components from the bottom up:

- ▶ Enterprise backbone: Infrastructure assets that are associated with your existing and running applications that support business operations.
- ▶ Reusable software components: Reusable assets, such as recipes, software patterns, and models, that can accelerate the development of SOA solutions.
- ▶ Business services: The key organizing principles that drive the design of IT to be aligned with business needs.
- ▶ BPM: Business Process Management is a key business initiative that enables companies to align strategic and operational objectives with business activities to fully manage performance through better informed decision making and action.

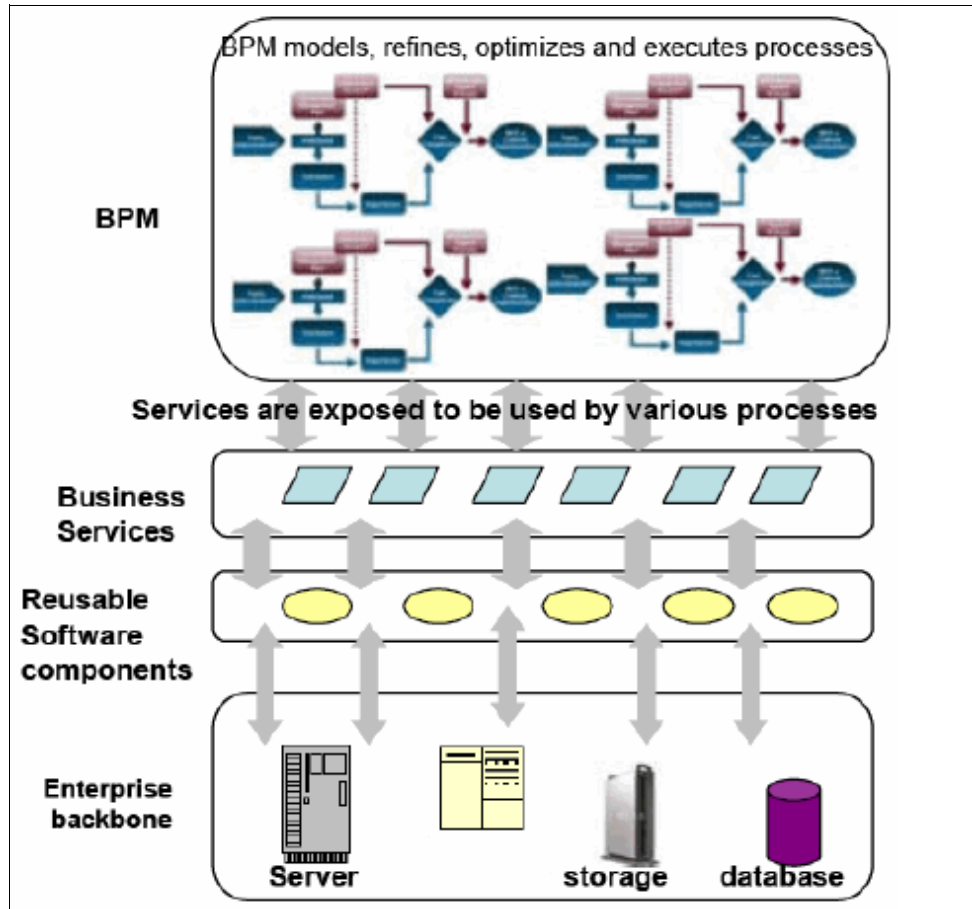


Figure 1-1 Four main components of enterprise architecture within the SOA framework

1.1.2 Major roles and activities in a SOA

There are three basic contributors to SOA solutions:

- Service providers** Publishes services and makes them available to clients. When IMS presents responses for resources from requestors, then IMS is a service provider.
- Service brokers** Brokers bring requestors and providers together by providing a registry of known services or Universal Description Discovery and Integration (UDDIs), along with the service contract that acts as the interface so that requestors can pick the correct service.
- Service requestors** These are the requestors of services. A requestor is typically a software application or service that requires a service. Requestors, such as IMS, make requests of service providers when it uses callout functions, which we describe throughout this book.

There are three actions that these contributors perform:

- Publish** A service provider registers with a service broker by publishing its supported interfaces.

- Discovery** Requestors initiate service requests by contacting the service broker and use directory services to discover a service that has a specific interface contract.
- Bind and Execute** A requestor uses information obtained by a service broker about a service provider to bind to a particular service and send a request to execute that service.

Figure 1-2 presents the major roles and activities that are aligned with SOA.

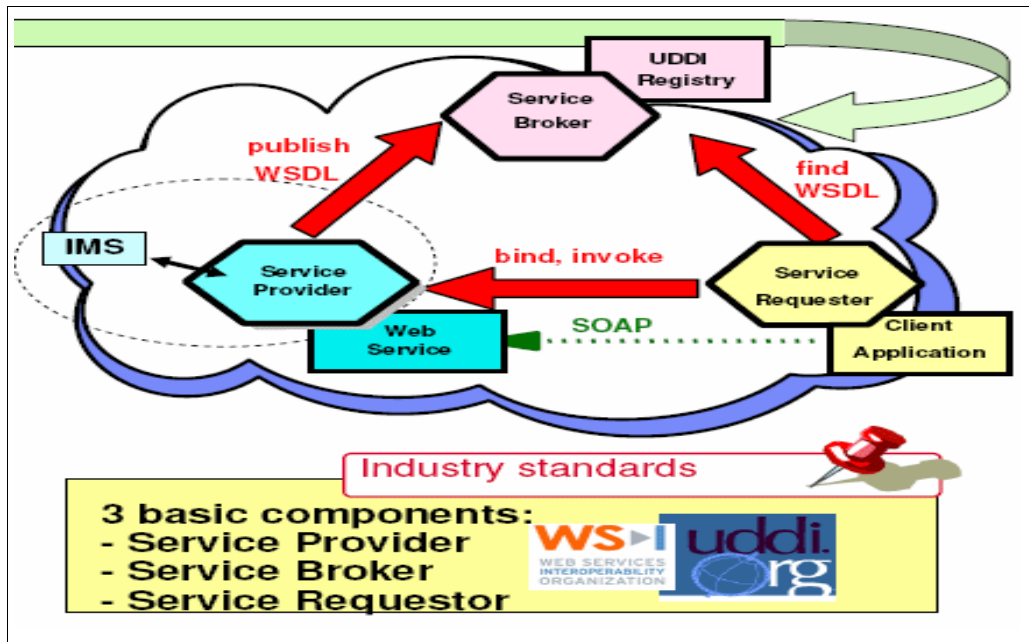


Figure 1-2 Major SOA roles and activities

1.1.3 SOA and standards

Figure 1-3 on page 7 helps to show how SOA is positioned as a service using defined standards.

SOA and Standards

- SOA is often defined as services exposed using the Web Services Protocol Stack

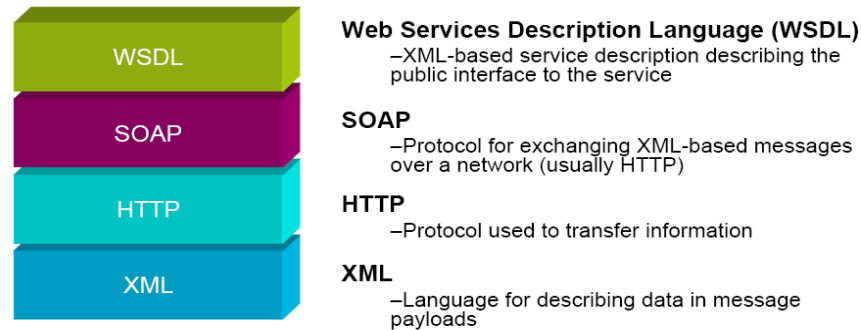


Figure 1-3 SOA exposure to the Web Services Protocol Stack

It is important that you understand the key SOA terminology that we mention in this book; therefore, in the next section, we define them.

1.2 Terminology

In this section, we define the meanings behind often used SOA related terminologies:

A feed	XML data that conforms to a specific type of data format. Feeds are used for content that is updated frequently and are read by feed readers.
A feed mashup	A feed mashup is a feed that is created by taking one or more source feeds and applying operators and functions to filter and restructure the source data.
WS-BPEL	Business Process Execution Language for Web Services (WS-BPEL) is the de facto industry standard for implementing processes on computing platforms. It is an XML-based language that allows you to define the logical flow of execution of services that constitutes a business process.
Component business modeling	Is a technique that IBM Global Business Services developed to help clients understand their business, the capabilities of the business, and identify capability gaps. It breaks the business down into independent areas to look for potential opportunities for improvements and innovation.

Note: For more information about Component Business Modeling, refer to the following resources:

http://www.ibm.com/services/us/gbs/bus/html/bcs_componentmodeling.html

<http://www.ibm.com/services/us/imc/pdf/g510-6163-component-business-models.pdf>

http://www.ibm.com/industries/financialservices/doc/content/bin/fss_bae_component_business_modeling.pdf

Enterprise software architecture	The sum total of all service-oriented software in the organization.
Java EE	<p>Java Platform, Enterprise Edition (Java EE) builds on the foundation of Java Platform, Standard Edition (Java SE) and is the industry standard for implementing enterprise-class service-oriented architecture (SOA) and next-generation Web applications.</p> <p>The name of the Java platform for the enterprise was simplified. Formerly, the platform was known as Java 2 Platform, Enterprise Edition (J2EE™), and specific versions had dot numbers, such as J2EE 1.4. The 2 and the dot number are dropped from the name. So the latest version of the Java platform for the enterprise is Java Platform, Enterprise Edition 5 (Java EE 5).</p>
Portlet	A portlet is a component of a portal Web site that provides access to some specific information source or application, such as news updates, technical support, or an e-mail program among many other possibilities. Portals aggregate different content into a single interface, and portlets connect the user to specific content within that interface. Most portals offer a selection of portlets that the user can select for a customized interface. Portals connect to portlets through portlet application programming interfaces (APIs).
Service	A service is a discoverable software resource that executes a repeatable task and is described by an externalized service specification.
Service component	An individual atomic piece of service-oriented software design (at least atomic from an architecturally significant point-of-view).
SCA	Service Component Architecture (SCA) is a set of specifications that describe a model for building applications and systems using a service-oriented architecture. SCA extends and complements prior approaches to implementing services and builds on open standards, such as Web Services.
Service Provider	A provider or implementer of a service (or set of services).
SO	Service-oriented (SO) system is a set of service-oriented software that is assembled to form a composite application. Its parts are called SO system parts that are created from service-oriented parts (which are individual pieces of service-oriented software that is used in multiple SO systems). SO parts come in two flavors: service consumers and service providers.
SOA	Service-oriented architecture is a business-centric IT architectural approach that supports integrating your business as linked, repeatable business tasks or services.
SOAP	Simple Object Access Protocol (SOAP) is an XML-based standard that defines how to move XML messages from point A to point B.

SOAP is protocol independent. It can be used over Transmission Control Protocol (TCP), Hypertext Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), or even Microsoft® Message Queuing (MSMQ).

SOAP allows for any number of message exchange patterns (MEPs), of which request/response is just one. Other examples include solicit/response (the reverse of request/response), notifications, and long running peer-to-peer conversations.

SOMA Service-Oriented Modeling and Architecture is an end-to-end software development methodology that IBM Global Business Services developed for building SOA solutions. It consists of identification, specification, realization, implementation, deployment, and management phases in which the fundamental building blocks of SOA are identified and refined and implemented in each phase.

TCP/IP Transmission Control Protocol/Internet Protocol (TCP/IP) is the basic communication language or protocol of the Internet.

UDDI protocol Universal Description Discovery and Integration (UDDI) is a directory model for Web Services. UDDI is a specification for maintaining standardized directories of information about Web Services, recording their capabilities, location, and requirements in a universally recognized format. It is considered, along with SOAP and WSDL, as one of the three foundation standards of Web Services.

Web Service A software component (callable piece of code) that is accessible (described, located and published) through standard network protocols, such as SOAP over HTTP. Web Services allow different applications from different sources to communicate with each other without time-consuming custom coding. Because all communication is in XML, Web Services are not tied to any one operating system or programming language.

Web Service is a powerful concept and is central in SOA implementations.

WSDL Web Service Description Language (WSDL) is an Extensible Markup Language XML based standard that contains a formal description of a Web Service. It describes the operations performed by the Web Service, the messages used by the Web Service, the data types, and the communication protocols.

In the WSDL we have three levels of descriptions, as presented in Figure 1-4 on page 10:

Abstract definition of the transaction/interface and input/output message layout.

Protocol binding (also called the Enterprise Information System (EIS) binding).

Service point that consists of TCP/IP addresses and URL details.

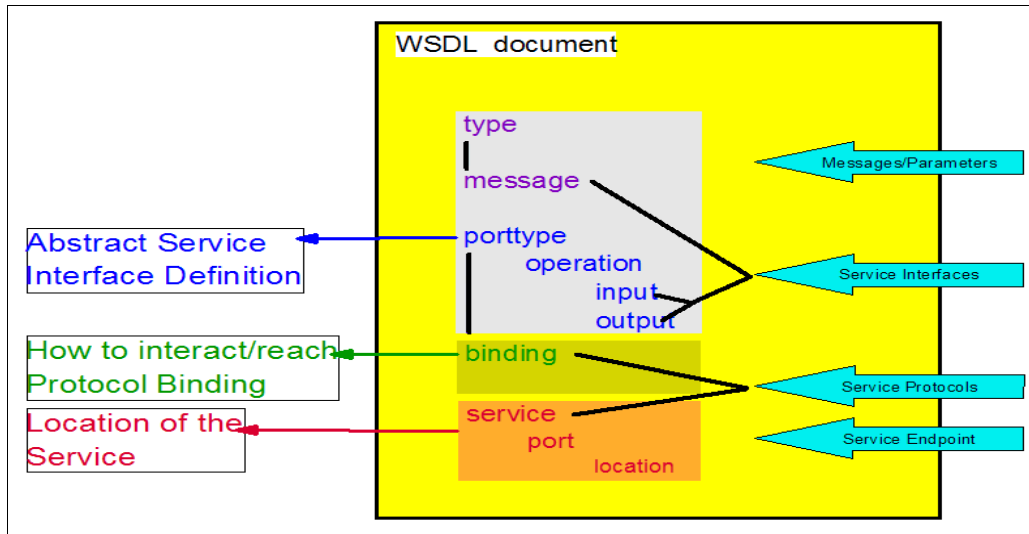


Figure 1-4 Three level of descriptions in a WSDL

WSDL: WSDL is based on XML tokens, so it is an XML document. WSDL is not only used for Web Services binding, but can also represent all kinds of bindings, even direct calls and invokes.

Most WSDL packages generate the WSDL file for you. However, it is useful to understand the file structure in case you might need to edit the file manually.

Example 1-1 illustrates a sample WSDL file structure.

Example 1-1 Sample WSDL file structure

```

<definitions>
<types>
  definition of types.....
</types>
<message>
  definition of a message....
</message>
<portType>
  definition of a port.....
</portType>
<binding>
  definition of a binding....
</binding>
</definitions>

```

WSRP

Web Services for Remote Portlets (WSRP) is a specification that defines how to leverage SOAP-based Web Services that generate mark-up fragments within a portal application. By defining a set of common interfaces, WSRP allows portals to display remotely-running portlets inside their pages without requiring any additional programming by the portal developers. To the end-user, it appears that the portlet is running locally within their portal, but in reality the portlet resides in a remotely running portlet container, and interaction occurs through the exchange of SOAP messages.

Leveraging WSRP within a service-oriented architecture provides a powerful combination whereby presentation-oriented portlet applications can be discovered and reused without engaging in additional development or deployment activities.

WS-Inspection

The Web Service Inspection Language (WS-Inspection) specification provides an XML format for assisting in the inspection of a site for available services and a set of rules for how inspection-related information must be made available for consumption.

A WS-Inspection document provides a means for aggregating references to pre-existing service description documents which were authored in any number of formats. These inspection documents are then made available at the point-of-offering for the service and through references that can be placed within a content medium, such as HTML.

XQuery

XQuery is a query language (with some programming language features) that is designed to query collections of XML data. It is semantically similar to SQL. XQuery can be used to easily and efficiently access the data sources involved and to provide responses in the form of XML messages. Example 1-2 illustrates a simple XQuery retrieving time-off details for the month.

Example 1-2 Simple XQuery

```
declare variable $employeeEmail as xs:string external;
declare variable $month as xs:string external;

let $employeeData := collection("emp.dbo.personnel")/personnel[emp_email =
$employeeEmail]
return
  <user name="{ $employeeData/emp_name }">
    <timeoff month="{ $month }"> {
      for $monthData in collection("emp.dbo.timeoff")/timeoff[emp_id =
$employeeData/emp_id and month_id = $month]
      return
        <event type="{ $monthData/type }" hours="{ $monthData/hours }"/>
    } </timeoff>
  </user>
```

The IT community is on its way to benefit from these reusable services and standardized interfaces. In the next section, we review how IMS fits in this new architectural style.

1.3 The value of including existing IMS assets into SOA

There is great value in utilizing existing IMS assets in modernization projects:

- ▶ Existing IMS assets support core business processes and provide crucial information.

Existing business processes in many cases are supported by heritage systems. Modeling business processes helps you identify the points of integration and interface. IMS can help you leverage running back-end processes and provide widely adopted integration points to support new business requirements.

- ▶ Existing IMS assets contain billions of lines of valuable business rules.
It is estimated that there are 200 billion lines of COBOL code in existence and replacement of heritage code is economically unfeasible and results in loss of valuable business information and rules.
- ▶ Using proven, time-tested IMS applications can significantly lower risk, cost, and time to market.
- ▶ The quality of the IMS system is recognized by all IMS users as fast, reliable, and mature.
- ▶ An IMS server is not a single point-of-failure. IMS has built-in recovery features, with take-over mechanisms, and supports a Sysplex environment.

We must concede that traditional IMS application development does not lend itself to a rapid market driven responsiveness model. Figure 1-5 presents an interesting statistic that is related to the number of discrete steps that are required to introduce a new IMS application in the traditional manner.

Responding faster and cheaper

- The real point behind SOA is in making it easier and faster to respond to changes in the marketplace
- If IMS application development or IMS database creation/modification is difficult and time consuming we cannot say IMS is truly keeping up with the SOA movement
 - A New IMS Application (written from scratch) takes up to 33 discrete steps
 - Requires the use of various different tools
 - JCL, TSO SPOC, Text Editors, FTP, Paper!!
 - Offers very little assistance and error checking

Figure 1-5 Speed to market for with traditional IMS application development

Considering IMS assets, we must distinguish two areas IMSDC (or IMS TM) and IMSDB:

- ▶ IMS Transaction Manager (TM): When only the TM component of IMS is utilized it is referred to as data communication controller (DCCTL), which is the facility to link applications that are running as transactions to networks.
- ▶ IMS Database Manager (IMSDB) or Data Base Controller (DBCTL): DBCTL interfaces with non-IMS communication controllers and traditionally supplies database services though the facilities of Database Resource Adapter (DRA) code.

Figure 1-6 on page 13 illustrates these two environments.

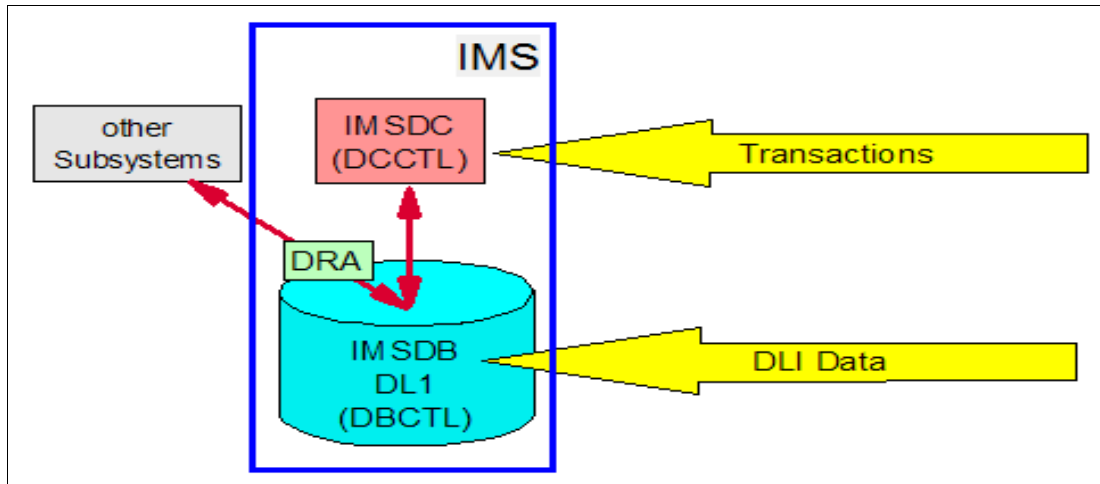


Figure 1-6 The IMSDC and IMSDB components of IMS

As a consequence, when considering IMS integration in SOA, we must consider both domains, TM and DB.

IBM recognized that it is necessary to provide the ability to leverage existing IMS transactions and make them available as callable Web Services. Figure 1-7 illustrates what was accomplished to this point. You can view the solutions as allowing network access to and from the IMS host environment and also opening IMS databases for access by non-IMS service requestors.

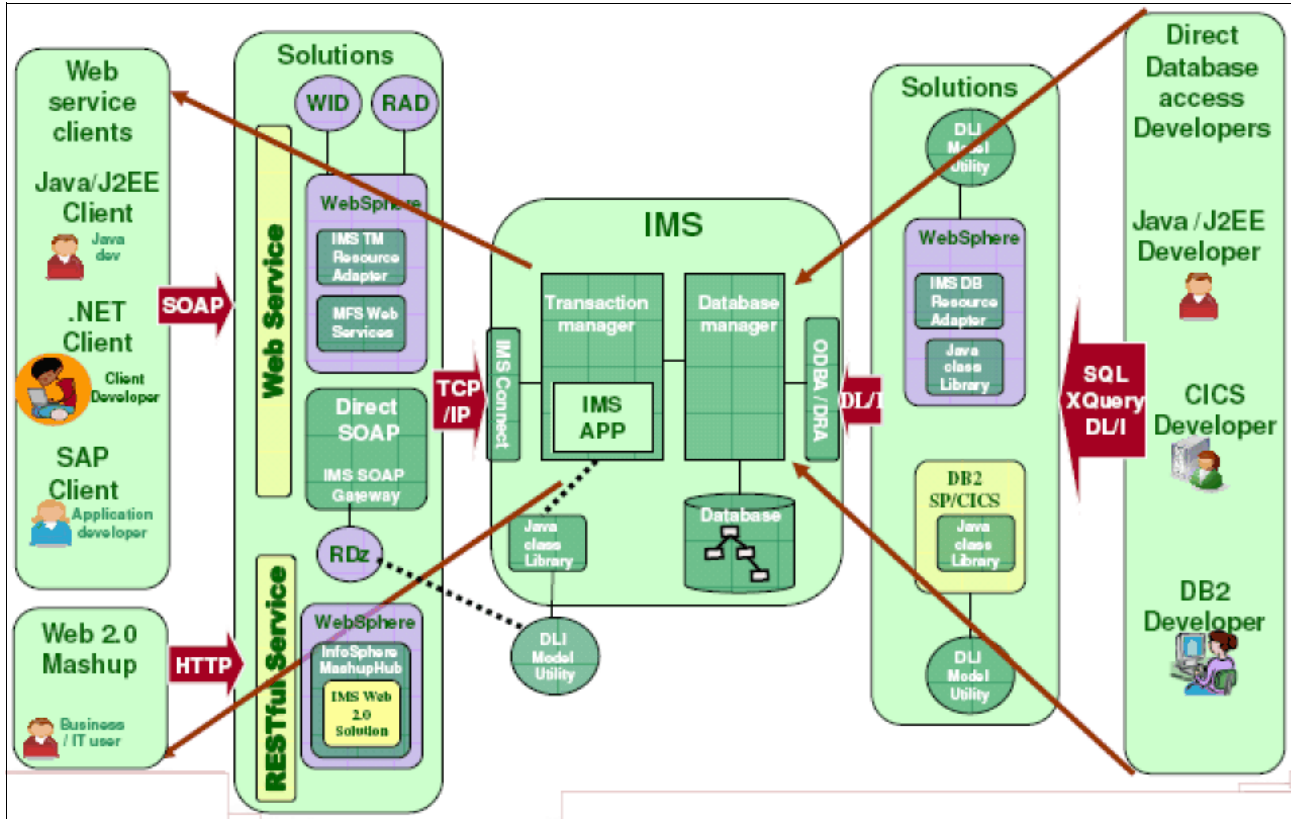


Figure 1-7 IMS and SOA: The big picture

We explain the components in this figure throughout this publication, but as a quick introduction we can highlight them now:

- ▶ IMS Connect is the TCP/IP gateway to IMS and runs within a separate z/OS address space to the IMS control region.
- ▶ The IMS TM Resource Adapter (TMRA) is a Java EE Connector Architecture (JCA) resource adapter that offers support to access existing IMS transactions from callable Web Services, Enterprise JavaBeans™ (EJBs), or even from HTML pages and servlets. IMS TMRA is a WebSphere Application Server-based solution. The tooling used in this solution is either IBM Rational Application Developer (RAD) or WebSphere Integration Developer (WID), and they receive definitions of the input/output messages in either COBOL, C, or PL/I and generate all of the necessary artifacts and code.
- ▶ Another WebSphere Application Server-based solution is targeted at applications that are MFS based. MFS is much more complex than, for example a simple COBOL copybook, and as such IMS offers a specific solution for MFS. This solution offers tooling that consumes the MFS definition of the input/output messages and again generates the necessary artifacts. The MFS solution is an IMS TM Resource Adapter client so it leverages all of its functionality.
- ▶ Another solution offered is the IMS SOAP gateway, which offers direct SOAP access to existing IMS transactions. It is a non-WebSphere based solution and does not require a Java EE (Java Platform, Enterprise Edition) container. There is tooling that is specific to this solution in RAD for z that consumes input/output message definitions, and once again generates all of the necessary artifacts for exposing an IMS transaction as a callable SOAP service.
- ▶ Although the TM-based solutions put focus on leveraging existing IMS transactions, the DB based solutions are focused on new IMS application development. These solutions offer direct IMS database access from a variety of environments. IMS allows WebSphere, CICS®, DB2®, and even application developers the ability to access IMS database assets using industry standard programming models and APIs.
- ▶ Two new IMS JMP and JBP-dependent regions were introduced. A JMP region is analogous to an MPP region and a JBP, to a non-message driven BMP region. The fundamental difference is that with the new regions, IMS now can fully house and maintain a JVM™ (Java Virtual Machine), which enables IMS to now effectively process Java workloads. In addition to this, IMS offers the Java class libraries, which contain a complete API for Java developers to use. The Java libraries offer a JDBC™ driver for IMS, which can process both SQL and XQuery expressions.

These same Java libraries can be utilized from several different runtime environments:

- WebSphere Application Server
 - DB2 stored procedures on z/OS
 - CICS using their JCICS API
- ▶ The IMS Open Database Access (ODBA) and DRA modules offer Java libraries the ability to access IMS databases from a non-IMS environment.
 - ▶ For the WebSphere Application Server environment, IMS offers another JCA resource adapter, the IMS DB Resource Adapter. The difference with this adapter, as opposed to the IMS TM Resource Adapter, is that all of the business logic is in the EJBs themselves. With the IMS DB Resource Adapter, there is no IMS-dependent region involved at all.
 - ▶ And for all of these solutions, IMS also offers tooling support through the DLIModel utility, which is an Eclipse-based GUI tool that offers visualization of IMS databases. It consumes PSB, DBD, and even COBOL copybook source to visualize all of the PCBs in a particular PSB. Information, such as hierarchies, segments, fields, and field types, are captured. With respect to new application development, the utility also generates database

metadata definitions that are consumed at runtime by the Java libraries, for example, this enables the libraries to convert a SQL query into a native IMS DLI call.

1.3.1 IMS Connect and IMS Connect Extensions

Because of the availability of IMS Version 9, IMS Connect was delivered as an integral component IMS. It performs a vital function as the connection from IMS to the TCP/IP world.

IMS Connect Extensions is a key tool for managing access to IMS through IMS Connect:

- ▶ Key benefits:
 - Provides event collection and instrumentation for IMS Connect
 - Streamlines operational management of IMS Connect and its clients
 - Assists in the development of TCP/IP clients and the transition to an SOA
- ▶ Principal users:
 - IMS tuning specialists, application developers, operators, and administrators

One example of using IMS Connect Extensions is in its assistance to monitor your IMS TCP/IP network flow. OMEGAMON® for IMS on z/OS, as a Real-time monitoring tool for IMS Connect, uses the Connect Extensions Publisher API where it obtains IMS Connect event records through the API. Figure 1-8 presents the output of a panel image from OMEGAMON for IMS on z/OS.

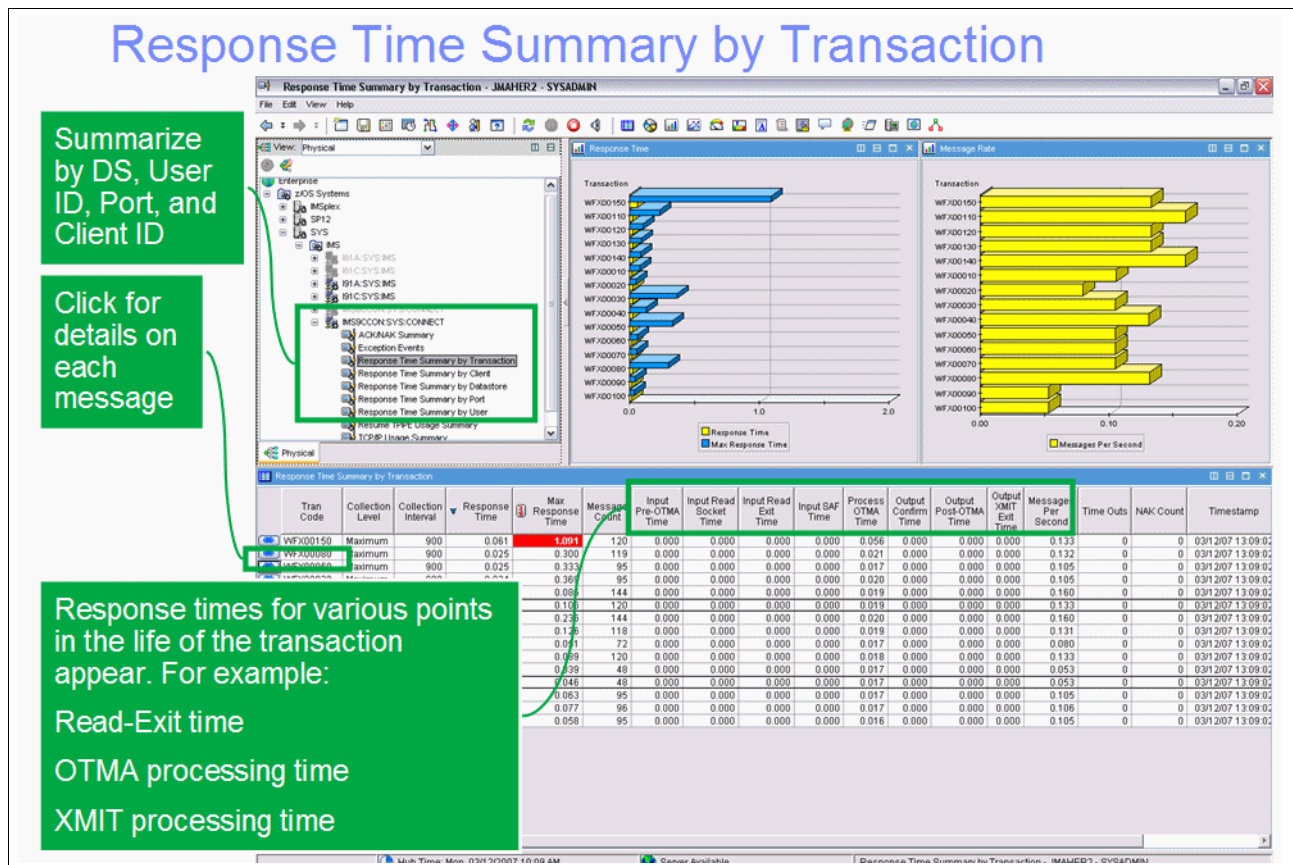


Figure 1-8 Omegamon use of IMS Connect Extension API records

To examine IMS Connect and IMS Connect Extensions more deeply, see Chapter 9, “IMS Connect and IMS Connect Extensions” on page 167.

In the next section, we introduce the IMS SOA Integration Suite and discuss how these facilities support IMS SOA solutions.

1.3.2 The IMS SOA Integration Suite

The IMS SOA Integration Suite leverages the utilization of your existing assets and running systems to integrate IMS capabilities in a SOA environment by providing these capabilities:

- ▶ Provides access to IMS transactions and data from any Web connection.
- ▶ Modernizes your IMS applications and enables them to operate with other clients, such as Microsoft.NET or SAP® clients in a service-oriented architecture.
- ▶ Integrates business logic that is embedded in your existing IMS applications with other IT systems, both within your enterprise and in the supply chain.
- ▶ Improves development time by using Java, instead of PL/I, COBOL, or Assembler.
- ▶ Accesses your IMS data directly for use by your applications from environments, such as DB2, CICS, and WebSphere Application Server.
- ▶ Stores and retrieves your XML content directly in IMS without any intermediate steps, and exchanges data with other systems by using established schemas.

The following tools and functions support access to IMS transactions:

- ▶ IMS SOAP Gateway
- ▶ IMS TM Resource Adapter
- ▶ IMS MFS Web Solutions
- ▶ IMS Web 2.0 Solution
- ▶ DLIModel utility and IMS XML DB

The IMS SOA Integration Suite enables you to access IMS transactions and data. In the next section, we provide an overview of these technologies and their business value.

1.3.3 IMS SOAP Gateway

IMS SOAP Gateway is an XML-based connectivity solution that enables existing or new IMS applications to communicate outside of the IMS environment using SOAP message protocol to provide and request services independently of platform, environment, application language, or programming model.

Figure 1-9 on page 17 illustrates the IMS SOAP Gateway deployment and runtime environment. IMS SOAP Gateway uses RDz to generate both the correlator and WSDL files that are used when deploying Web Services within IMS SOAP Gateway. An IMS SOAP Gateway Deployment utility is included for you to set up properties to deploy and maintain IMS Web services. Also, an IMS SOAP Gateway Administrative Console is available to list the deployed Web services when the server is started.

IMS SOAP Gateway interfaces with IMS Connect using TCP/IP protocols but uses SOAP or HTTP or HTTPs when communicating with SOAP Clients.

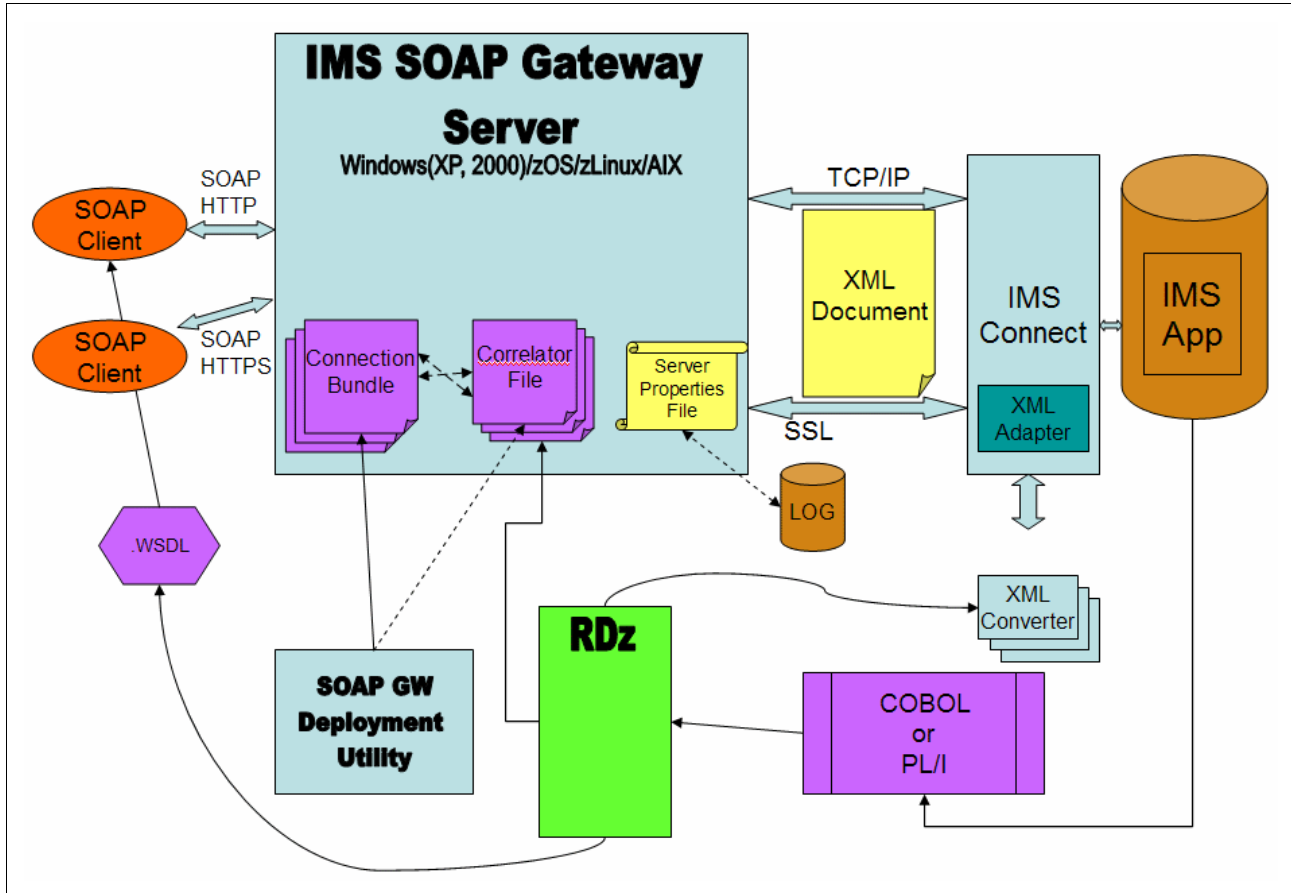


Figure 1-9 IMS SOAP Gateway development and runtime environment

The value of using SOAP Gateway

IMS SOAP Gateway is a light-weight Web Services solution that enables IMS applications to interoperate in a SOA environment without needing a full-blown application server (for example, Java EE Server). One typical usage scenario of providing Web Services with the IMS SOAP Gateway is to enable Microsoft.NET client applications or intermediary servers that submit SOAP requests into IMS to drive business logic transactions.

Through SOAP, IMS SOAP Gateway provides and requests services that are independent of platform, environment, application language, or programming model:

- ▶ It enables IMS application assets as Web Services.
- ▶ It allows non-WebSphere customers to reuse existing and to create new IMS-based business logic.
- ▶ It operates with any types of client application using SOAP/HTTP protocols.

Generated IMS service definitions (that is Web Services Description Language files) can be published or exposed to an UDDI directory for businesses to publish their offerings and for users to discover their needs. You can retrieve IMS WSDL files out of the UDDI directory and fit them into a tool (such as Microsoft.Net or Apache Axis server tools) to generate SOAP messages to be sent to the host to run existing IMS applications.

Product support information:

Table 1-1 summarizes the IMS SOAP Gateway versions.

Table 1-1 SOAP Gateway versions

IMS SOAP Gateway version	IMS and IMS Connect	IDE (Integrated Development Environment)
9.2.1	Version 9 and 10	Rational Developer for System z Version 6 and 7 (formerly known as WebSphere Developer for z)
10.1	Version 9 and 10	Rational Developer for System z Version 7.1.1

The Program Identification Number (PID) for SOAP Gateway is 5655-R04. For a detailed discussion about IMS SOAP Gateway, refer to Chapter 10, “The IMS SOAP Gateway” on page 203.

1.3.4 IMS TM Resource Adapter

The IMS TM Resource Adapter, previously known as IMS Connector for Java, is part of the IMS SOA Integration Suite of middleware functions and tools. Using the IMS TM Resource Adapter you can quickly and easily create Java applications that access new and existing IMS transactions over the Internet. Using TMRA within a WebSphere or Rational-family development environment, you can:

- ▶ Develop components of business processes in support of SOA.
- ▶ Create Java EE applications from Java beans.

The development version of the IMS TM Resource Adapter is included in the following integrated development environments:

- ▶ Rational Application Developer for WebSphere Software
- ▶ WebSphere Integration Developer
- ▶ WebSphere Transformation Extender
- ▶ Rational Developer for System z (formerly known as WebSphere Developer for System z)
- ▶ Rational Software Architect

You can download the runtime component of the IMS TM Resource Adapter from this Web site:

<http://www-01.ibm.com/software/data/ims/ims/components/tm-resource-adapter.html#downloads>

The value of using IMS TMRA

The IMS TM Resource Adapter implements the Java EE Connector Architecture (J2C), which connects Enterprise Information Systems (EISs), such as IMS to the Java EE platform. The Java EE Connector Architecture provides your applications with the qualities of service that a Java EE application server can provide, such as connection, transaction, and security management, which allows for:

- ▶ You can use the IMS TM Resource Adapter with a Java EE server, such as IBM WebSphere Application Server when a Java application accesses an IMS transaction that is running on a host IMS system. The IMS TM Resource Adapter also enables an IMS application to act as a client to invoke applications in a Java EE server.
- ▶ Although the IMS TM Resource Adapter is intended for use primarily by Java applications or Web Services that submit transactions to IMS, the IMS TM Resource Adapter can also be used by services that submit IMS commands to IMS.

Figure 1-10 maps the IMS TMRA deployment and runtime environment.

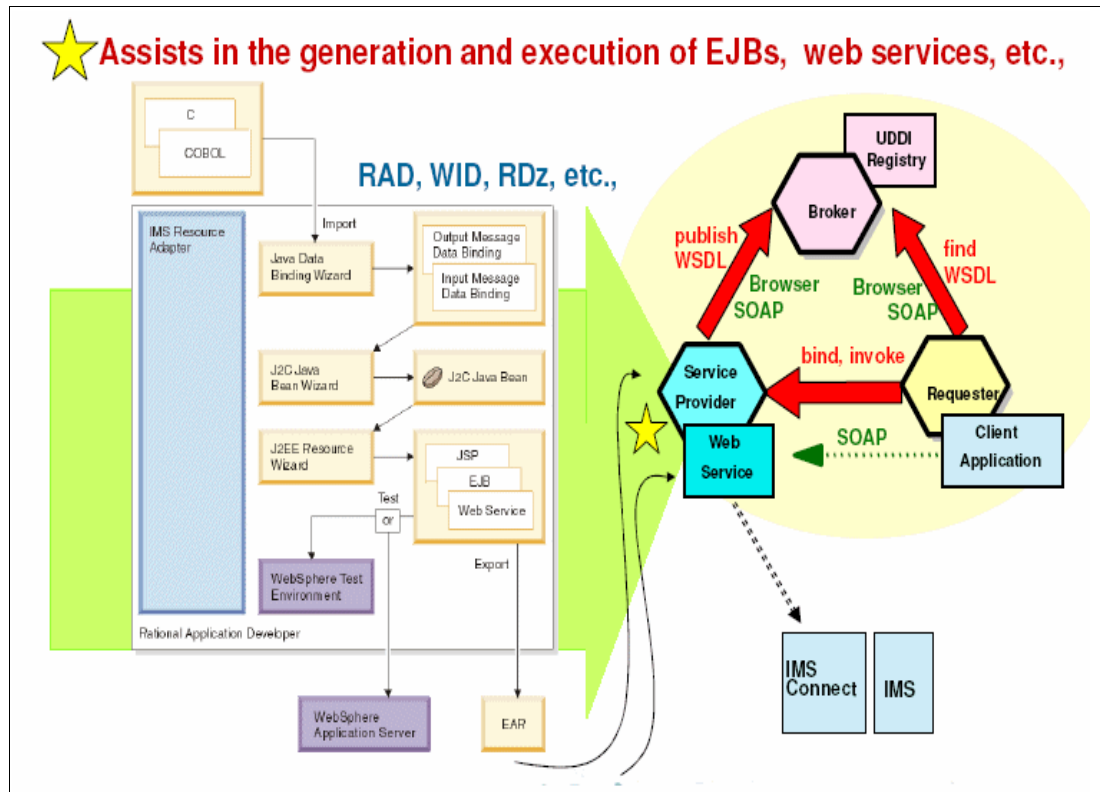


Figure 1-10 IMS TM Resource Adapter development and runtime environment

In Figure 1-10, we see that TMRA provides tooling support for development of Java EE applications, Web Services, and business processes that access IMS transactions in various Rational and WebSphere-integrated development environments.

TMRA also provides programming for deployment to the WebSphere Application Server and WebSphere Process Server (WPS) runtime environment on many platforms, which includes z/OS and Linux® on System z.

IMS TM Resource Adapter Version 9 has PID number 5655-J38, and IMS TM Resource Adapter Version 10 uses PID number program number 5635-A01. For more details, review Chapter 11, “The IMS TM Resource Adapter” on page 225.

1.3.5 IMS MFS Web Solutions: Web Enablement, services, and SOA support

MFS Web Solutions provides the tooling utility and runtime support to Web-enable existing or new IMS MFS-based applications in the IBM WebSphere Application Server and interactively render them for display in standard browsers, such as Microsoft Internet Explorer® and Mozilla Firefox. IMS MFS Web Solutions consist of IMS MFS Web Enablement, IMS MFS Web services, and IMS MFS SOA support.

The value of using IMS MFS Web Enablement

MFS Web Enablement support provides Business-to-Client (B2C) solutions to reuse customers’ existing MFS-based IMS applications. The enhancements target the market by

supporting MFS Double Byte Character Set (DBCS) and Extended Graphics Character Set (EGCS) and providing security and accessibility updates.

It also addresses the following requirements:

- ▶ Input LTNAME support: Customers use MFS screens and conversations extensively today. They must access IMS applications that use RACF® and Logical Terminal Name (LTNAME) for authentication and authorization using MFS Web enablement.
- ▶ Accessibility compliance update: The MFS Web Enablement sample CSS (Cascading Style Sheets) stylesheet is updated to generate accessible Web pages by default. Users can customize the CSS stylesheet.

Figure 1-11 is an overview of the IMS MFS Web Enablement component. It is housed within a WebSphere Application Server and uses the support of IMS TMRA to communicate with IMS Connect.

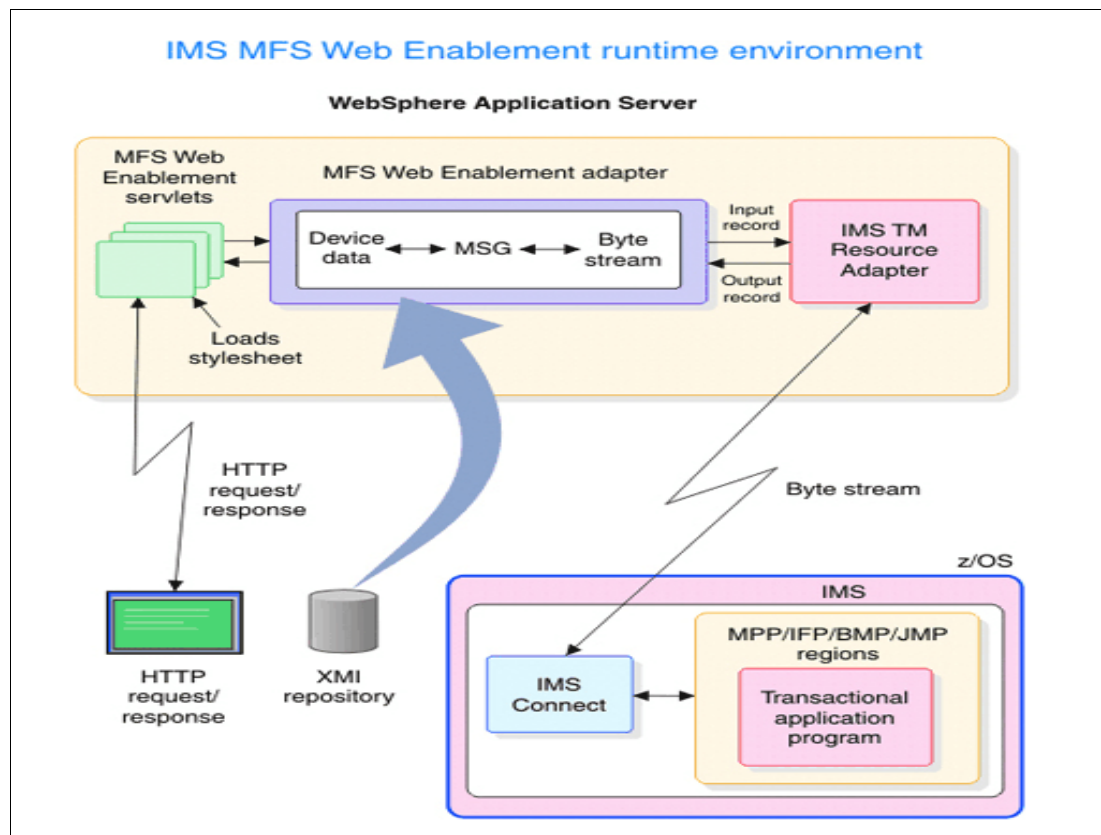


Figure 1-11 The IMS MFS Web Enablement runtime environment

The value of using IMS MFS Web services

IMS MFS Web services are superseded by IMS MFS SOA support, which is the next generation of MFS support.

IMS MFS Web services support enables you to reuse existing MFS-based IMS business logic as Web services in the WebSphere Application Server. The MFS Web services tooling component, also known as the MFS Importer, is part of the IBM WebSphere Studio Application Developer Integration Edition. The corresponding runtime component of MFS Web services is included in IMS TM Resource Adapter, on both distributed and z/OS platforms.

The value of using IMS MFS SOA support

The MFS SOA support meets customer requirements to provide a continued systematic approach of transforming existing MFS-based IMS applications (including conversational transactions) into Web Services.

Existing MFS Web services customers must use the new programming model supported in RAD/WID/WDz. IMS MFS SOA support is integrated with the Enterprise Metadata Discovery (EMD) framework in Rational Application Developer Version 7.5 or later to transform existing MFS-based IMS applications into J2C Java beans and J2C Java Data Binding classes. After you create a J2C Java bean, you can create Java EE artifacts, such as JSPs, EJBs, or Web services to deploy the generated J2C Java bean.

Figure 1-12 displays the methodology to obtain and compose MFS source to generate Web Services.

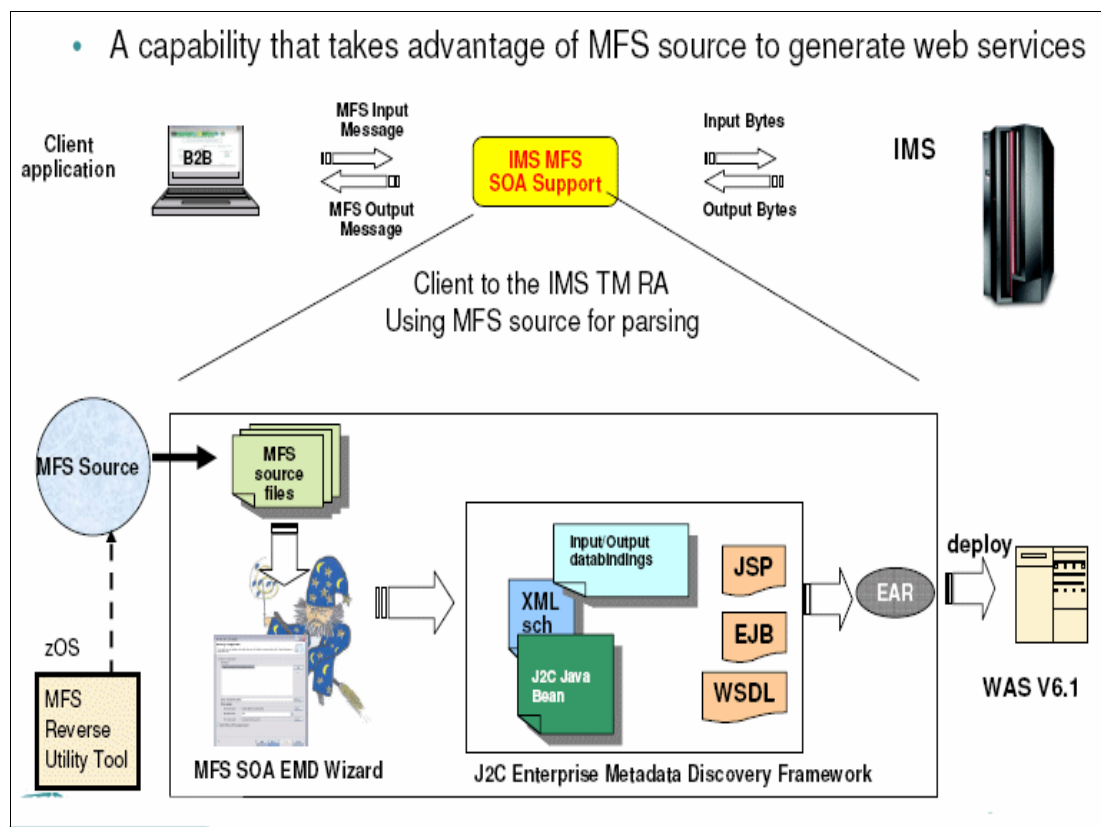


Figure 1-12 MFS Source to generate Web Services

See Chapter 12, “MFS Web solutions” on page 245 for more detail.

1.3.6 IMS Web 2.0 Solution: IMS Info 2.0 for InfoSphere MashupHub

IMS Info 2.0 is the IMS Web 2.0 solution that is embedded in InfoSphere™ MashupHub that unleashes your IMS assets into XML, Atom Syndication Format (ATOM), which is an XML language used for Web feeds) or RSS (Really Simple Syndication) feeds. RSS is a family of Web feed formats that are used to publish frequently updated works, such as blog entries, news headlines, audio, and video, in a standardized format. The Atom format was developed as an alternative to RSS.

InfoSphere MashupHub provides visual tools for creating, storing, transforming, and remixing feeds to be utilized in mashup and situational applications. It also offers a central catalog for users to tag, rate, and share assets. InfoSphere MashupHub is bundled in the IBM Mashup Center.

Mashups were rated by Gartner as Top 10 Strategic Technology for 2008¹.

“By 2010, Web mashups will be the dominant model (80 percent) for the creation of composite enterprise applications. Mashup technologies will evolve significantly over the next five years, and application leaders must take this evolution into account when evaluating the impact of mashups and in formulating an enterprise mashup strategy.”

The value of using IMS Web 2.0 Solution

With InfoSphere MashupHub's built-in IMS support enterprise data can be made available to the Web 2.0 community, and your users can further remix and mash up the data to meet their business needs.

In IBM Mashup Center Enterprise Edition Version 1.0, you can create ATOM feeds from IMS transactions. With the embedded IMS Info 2.0, you can:

- ▶ Create ATOM feeds from IMS transactions that run on IMS Version 10 with the integrated IMS Connect Version 10. Both COBOL and PL/I applications are supported.
- ▶ Easily customize IMS transactions without modifying the original application. Through a Web interface you can specify which input parameters to expose to users and the default parameter values to invoke the feed. Feeds can be further restructured and customized by using the operators and functions in InfoSphere MashupHub.
- ▶ Utilize the tooling support in Rational Developer for System z Version 7.1.1 to generate the required XML converter driver and correlator. The XML converter driver is used by IMS Connect to transform the data between XML and bytes. The correlator is used by IMS Info 2.0 to map the request and response messages to the input and output data structures of the IMS application.

Figure 1-13 on page 23 presents an overview of the use of the Web 2.0 components as applied to IMS.

¹ Reference is URL:
<http://www.gartner.com/it/page.jsp?id=530109>

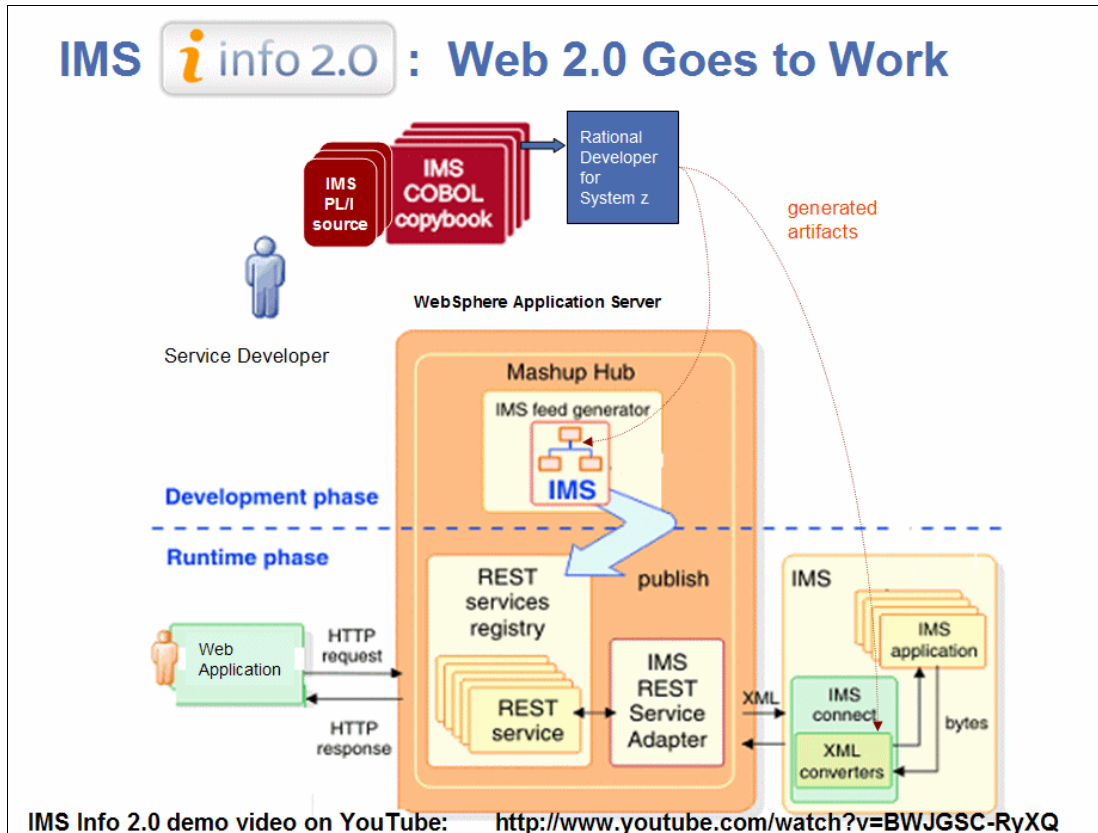


Figure 1-13 IMS info 2.0

For more details about IMS Web 2.0 solutions, see Chapter 13, “IMS Web 2.0 Solution” on page 271. You might also want to visit the IMS Info 2.0 demo video on YouTube:

<http://www.youtube.com/watch?v=BWJGSC-RyXQ>

1.3.7 The DLIModel utility

Without the DLIModel utility, the IMS-Java user must manually create Java classes that describe the metadata (for example, structure, segment layouts, and so on) of the IMS databases that are to be processed. The DLIModel utility creates these classes from PSB and DBD source plus optionally, high-level language source that more fully describes segment field layouts. The DLIModel utility supplies a Web download version that runs as a plug-in to Eclipse, WebSphere Developer for System z, RAD for WebSphere, and RDz.

The value of using the DLIModel utility

The business goal of the DLIModel utility is to ease IMS application development. Using the DLIModel utility you can transform your IMS database information (program specification blocks, database descriptions, and COBOL copybooks) into application-independent metadata. In addition to creating metadata, using the IMS DLIModel utility you can:

- ▶ Generate XML schemas of IMS databases, which are used to retrieve XML data from or store XML data in IMS databases.
- ▶ Incorporate additional field information from COBOL copybooks.
- ▶ Incorporate additional PCB, segment, and field information or override existing information.

- ▶ Generate a DLIModel report, which is designed to assist Java application programmers in developing applications based on existing IMS database structures.
- ▶ Generate an optional DLIModel trace log.

Figure 1-14 illustrates the input and output flows from the DLIModel utility.

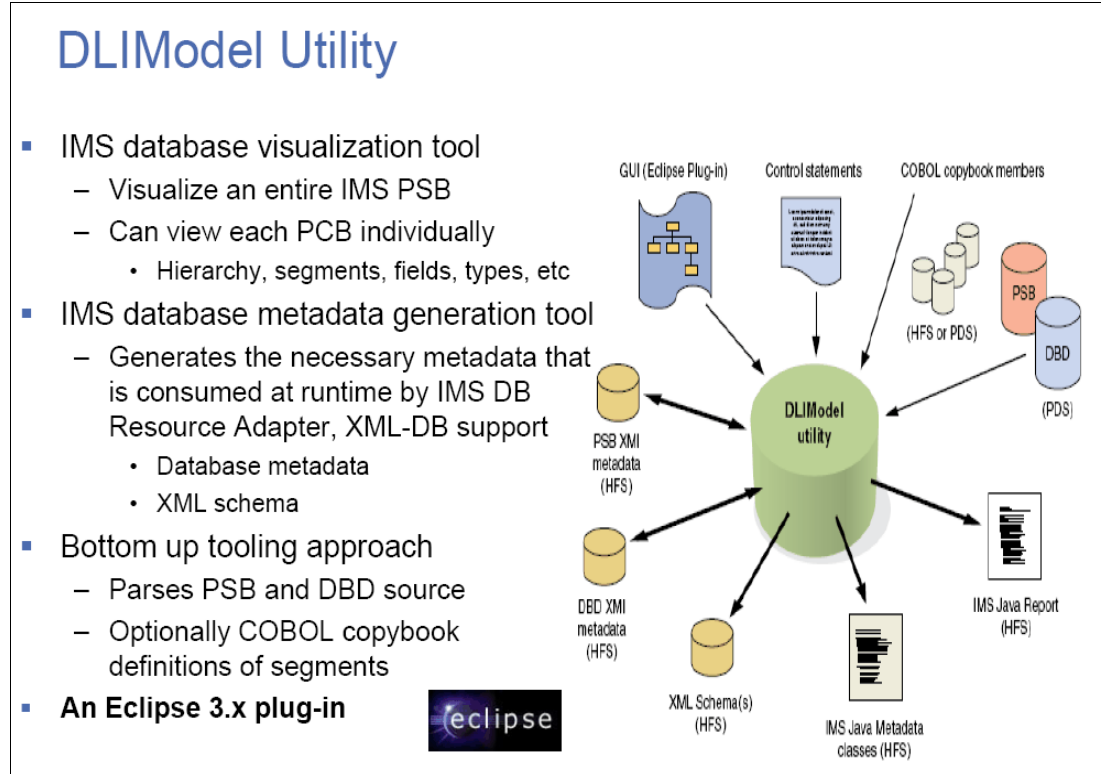


Figure 1-14 DLIModel Utility

A detailed description is in Chapter 14, “DLIModel Utility, DB Web Services, and XQuery” on page 293.

1.4 IMS Open Database

The IMS Open Database that was announced with IMS Version 11 provides distributed access to IMS database resources. It also helps to drive open standards and open technology into IMS. The open standards that are introduced into this solution include the Java EE Connector Architecture, JDBC, and DRDA®.

Historically, the IMS database was a closed architecture, and by opening it up, IMS is positioned for the future as it pertains to industry standard access APIs and the emerging SOA market.

The business challenges addressed by Open Database:

- ▶ Data can be difficult to access outside of the IMS environment. Clients might want to participate in the emerging SOA market.
- ▶ Traditional IMS support skills are becoming increasingly rare and difficult to acquire and train.
- ▶ It requires too long a time to deploy new applications and enhance existing applications.

The value of using IMS Open Database

The ability to access IMS data from outside of the IMS environment simplifies the process of developing new applications that leverage existing investment in IMS data. Also, it reduces costs by providing distributed access to IMS database resources through industry-standard interfaces.

The distributed access function offers two distinct types of IMS database resource distribution:

- ▶ The distribution of IMS database resources across LPARs in an IMSplex. IMS data can be accessed through TCP/IP from a Java EE application server that resides on a z/OS platform that is on a different logical partition (LPAR) from the IMS subsystem.
- ▶ Pure distribution, which means that IMS database resources are now directly accessible from non-mainframe platforms, which includes full distributed transaction processing and two-phase commit semantics.

All of this is accomplished by three main components:

- ▶ Client-side libraries that implement the industry-standards interfaces and protocols.
- ▶ IMS Connect, which processes the distributed requests.
- ▶ The use of the Open Database Manager (ODBM) address space.

Figure 1-15 introduces the Open database environment available with IMS Version 11.

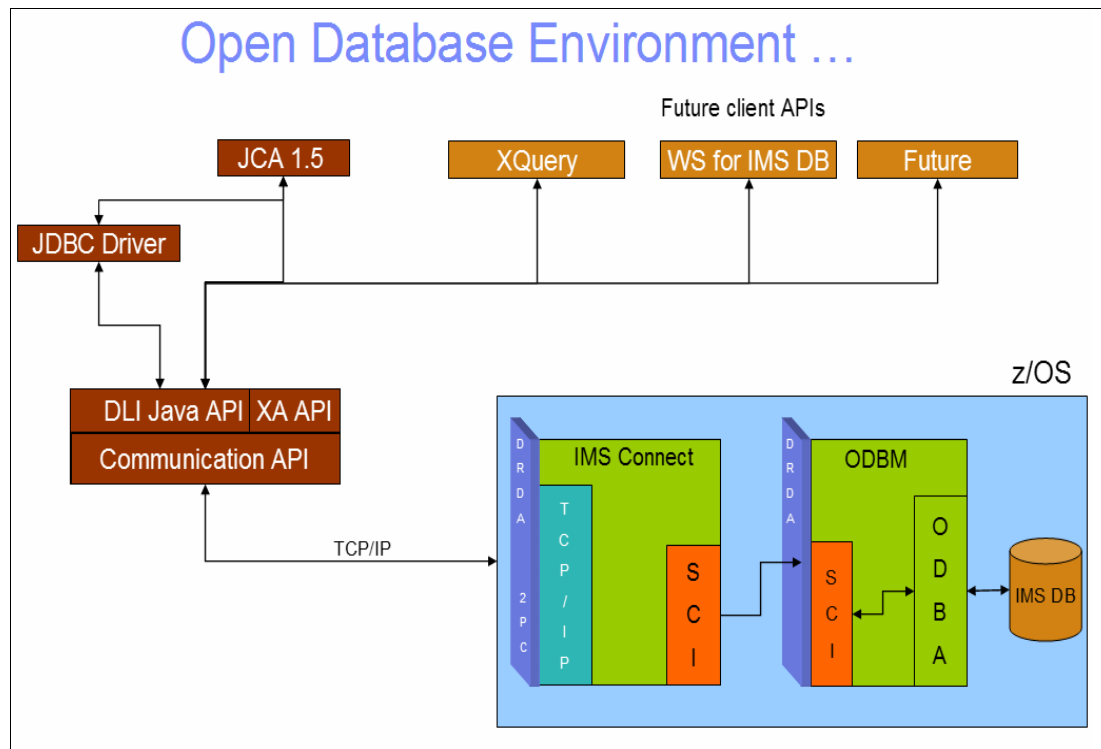


Figure 1-15 Open DB environment

For more details about Open Database, see Chapter 15, "IMS Open Database and Universal Drivers" on page 321.

1.5 DataPower and IMS

DataPower® is a powerful solution for XML acceleration, XML security, monitoring, and managing SOA environments.

1.5.1 The value of using DataPower

Included in the DataPower firmware V3.6.1 is a feature called "IMS Protocol Support" that adds support to allow Multi-Protocol Gateway services to accept IMS connections from clients and connect to IMS-based applications. This functionality provides:

- ▶ An IMS Connect proxy to IMS Connect clients. The use case is for existing IMS Connect clients who want to make in-flight modifications to headers and payloads without changing the client or IMS.
- ▶ Web Service facade to IMS Connect transactions. The use case is to make use of the strong Web Service features in DataPower to quickly enable Web Service support for IMS Connect.

Figure 1-16 presents the DataPower hardware components and their primary roles.

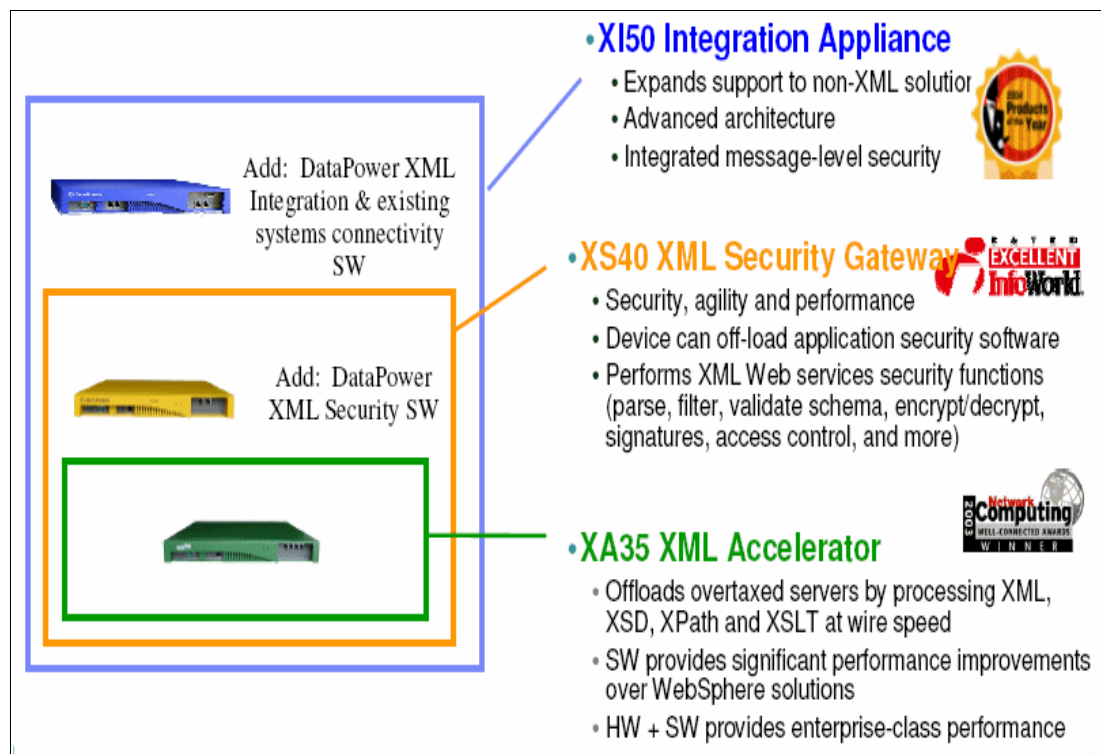


Figure 1-16 DataPower components

A more detailed description is in Chapter 16, "Using DataPower with IMS" on page 341.

This concludes our introductory chapter on IMS and SOA.



Development phases of IMS SOA projects

Many corporations invested millions of dollars worth of mainframe assets and core applications that support the heart of the enterprise. In fact, it is estimated that some five trillion USD worth of applications — critical business assets — reside on today's IBM mainframe systems and 60% of the world business data.

The convergence of SOA and mainframe technologies can liberate these core business assets by making it easier to enrich, modernize, extend, and reuse them well beyond their original scope of design. As a result, you can deliver new value more rapidly, affordably, and at a lower risk than by rebuilding and replacing what already works well for the business. That lets you streamline business processes and increase overall business flexibility.

In this chapter, we describe the business value of IMS application modernization and present suggested phases that are associated with IMS application integration into Enterprise SOA solutioning.

2.1 IMS as a SOA integration focal point

We stated that IMS is an integration focal point for Enterprise service-oriented architecture. The reasoning behind this follows:

- ▶ Business assets are product and platform independent and other applications and services can benefit from access to those assets. IMS holds the company's core business data, such as parts information, maintenance data, financial information, insurance policies, and inventory data. The logic contained in IMS transactions and batch programs is also a major core business asset.
- ▶ IMS Connect provides access from any platform and access from IMS SOAP Gateway and DataPower.
- ▶ IMS SOAP Gateway opens access from applications, such as SAP server, Microsoft .NET and Apache, and other distributed applications.
- ▶ IMS DB can be driven by IMS TM, CICS, DB2 stored procedures, WebSphere Application Server, and IMS Open Database Access (ODBA) applications.
- ▶ IMS host applications can access IMS DB (including IMS XML DB) and DB2 databases and communicate with WebSphere MQ for queue access. IMS supports JDBC access to IMS DB from both distributed and host applications.
- ▶ IMS host applications can access external applications (any platform) through IMS Connect. This might be an Enterprise Java Bean, Message Driven Bean (MDB), a Web Service, or other application.
- ▶ IMS host applications can be quickly and easily converted into Web Services using Rational Tools. Figure 2-1 on page 29 introduces how Rational Developer for System z can be used for both mainframe and client server development of many cross platform solutions.

Innovative Development: Assemble Services Across Platforms

Rational Developer for System z

- Bridges mainframe and client-server development
 - ▶ For end-to-end integrated processes running across platforms
 - ▶ Shared developer repositories and tooling framework
- For deployment to WebSphere, CICS, IMS, DB2 and batch
 - ▶ Cross platform development
 - ▶ Workstation based development tool
 - ▶ Workstation based mainframe application analysis tool
- Develop SOA systems that combine Web services, web applications and traditional applications
 - ▶ COBOL, PL/I, EGL (4GL)
 - ▶ Java, J2EE, WSDL, Java Beans, XML adapters
 - ▶ Portlets and complex user interfaces
 - ▶ Service flow modeler for CICS and HATS
- Wizards for generating code to be used in CICS and IMS integration

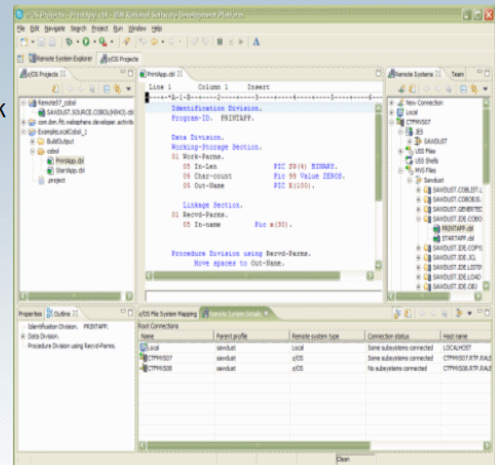


Figure 2-1 Innovative development of cross platform solutions

- ▶ New applications can be written in Java for distributed platforms without knowledge of the IMS DL/I application language, which enables organizations to utilize the existing skills of their development teams without needing to invest in other skill sets.
- ▶ Performance continues to be a high priority for Clients. IMS continues to exceed the highest performance needs of our customers while maintaining a low cost per transaction. Current benchmarks taken at the IBM Silicon Valley Lab in July 2008 resulted in the following statistics for transactions per second (tps):
 - IMS Shared Queues and Block Level Data Sharing environment: 21,396 tps
 - IMS Fast Path single image: 29,141 tps
 - IMS Connect environment: 14,200 tps
 - IMS / DB2/ IMS Connect / IMS Connect Extensions in a Block Level Data Sharing environment: 7,312 tps

Figure 2-2 on page 30 summarizes the benefits of including IMS in your SOA solutions.

IMS: The High-Performance Application Server for System z

- * Integration focal point for SOA
- * Remarkable performance
- * Rock-solid reliability and security
- * Most cost efficient run-time environment
- * Integrated message queuing, transaction processing and data base management
- * Open, standard interfaces allowing 'any-to-any' connectivity and access
- * Fully integrated into today's A/D toolsets
- * Natural XML support
- * Robust runtime support including JAVA, C, COBOL, PLI and Assembler

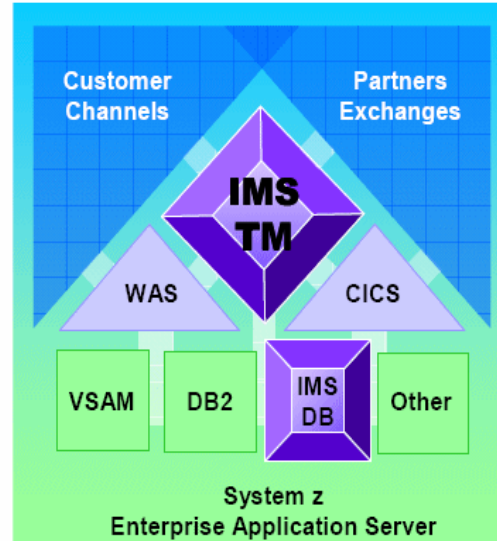


Figure 2-2 Benefits for the use of IMS in your SOA solutions

SOA offers a flexible and extensible approach to reusing and extending existing applications and services and constructing new ones. IBM will continue to invest heavily in IMS to enhance IMS to assist Customers transform their core business processes with emerging technologies using IMS in the SOA framework¹.

2.2 The value of IMS service reuse

IMS and the concept of reuse offer great value in a SOA environment. Using service orientation your development team can reuse components.

Traditionally to reuse a system component, developers had to understand how to interface with this component and integrate it in their system, which usually involved the integration developer to understand the technicalities and implementation of that component to perform a successful integration. In many cases, the process also included tweaking the component to integrate it into the new environment and performing some testing to make sure that the functionality of that component was not impacted because of the integration activities.

With service orientation, the way service components are invoked and their functionality defined is standardized. As we will see in the coming chapters, all the integration developer needs to know about a service component is how to invoke the service provided by that component and the location of that component. The integration process does not change the internal implementation of the component or its location. Service components are platform independent and can be invoked and consumed on any platform, which significantly reduces the time and effort in integration and testing.

¹ Business value of IMS and SOA: Better together July 2007, IMW11876-UUSEN-00

The same is true for IMS services, for example, an application developer can quickly compose a distributed application, reusing the services provided by IMS without having to know how these services are implemented. IMS services that were tested in production can be easily reused in developing new applications with the same level of confidence in performance and reliability.

Reduction in application development, maintenance, and testing effort is significant. Other benefits include:

- ▶ Reduction in the cost of maintaining application portfolios:
 - Applications are less complex because they are reusing a number of well understood and tested services.
 - Application developers only build the presentation layer.
 - Little additional skills are required because most modern application development tools support seamless invocation of services.
 - Changes in existing enterprise applications can be met with minimal effort and significantly reduced schedules. Implementing changes usually involves reconnecting existing services.
- ▶ New business requirements can be quickly and cheaply realized by reconnecting the services to realize the new requirements.
- ▶ Reusing IMS services leads to a reduction of business risk:
 - Errors are limited to new development, not in the reuse of existing services. Therefore there is the opportunity of less rework and testing because of less defects.
 - Tested services maintain the same performance levels and functionality.

Figure 2-3 summarizes some of the advantages of reusing services.

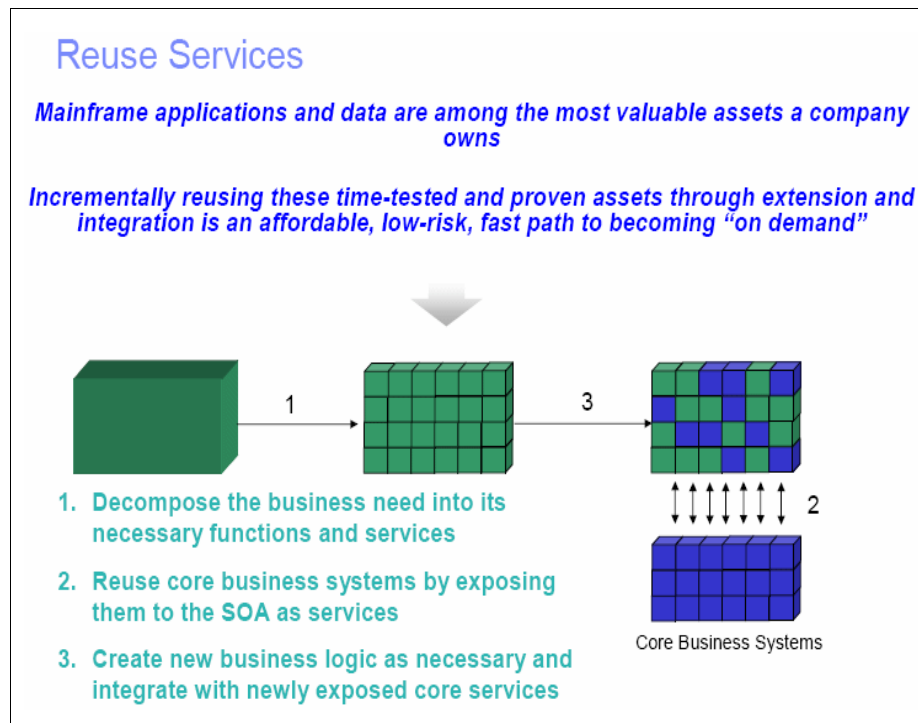


Figure 2-3 Reuse of application services

2.3 Styles of business transformations

Adopting SOA involves business transformations that impact business processes and performance, flexibility, and business value. To move towards an On Demand business, customers must transform their technical infrastructure from unique, single purpose applications, to shared resources. This transformation might keep the basic business processes unchanged. The styles of transformation might exploit existing enterprise assets. These styles of transformations are:

- ▶ **Improve:** Simply, this involves putting a new face on your existing application programs, for example, adding a Web interface to your existing 3270 green panel and keeping the applications intact.
- ▶ **Adapt:** Build a larger solution and leverage the existing applications and data. Using IMS you can bring the existing IMS data and transactions into the solution mix.
- ▶ **Innovate:** Applications are re-engineered to realize business needs.

Figure 2-4 presents these three styles in which business processes can be transformed.

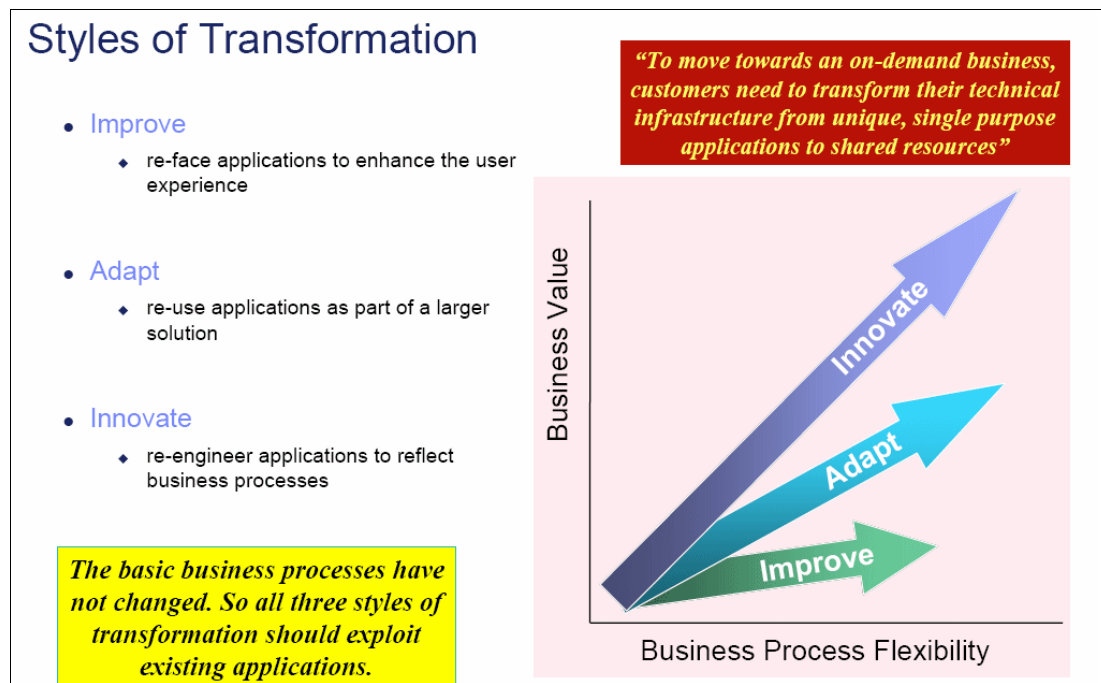


Figure 2-4 Styles of transformation of business processes

Regardless of the style of transformation that exist, a step-by-step plan of attack is required for successful implementation.

2.4 Methodology of the development of SOA projects

In this portion of this chapter, we present a methodology to engage in your SOA journey. The methodology is specifically customized to address integration with your existing running processes.

The size and duration of your project must not significantly affect the methodology you undertake because the methodology is generic to accommodate different project sizes and requirements.

The process defines the overriding principles that you must follow in each of your project phases. Depending on the size of your project, you can divide it into a number of phases, with each phase viewed as a project of its own.

In the next section, we take a closer look at the methodology.

2.4.1 The Rational Unified Process

The Rational Unified Process (RUP) is an iterative software development process framework created by the Rational Software Corporation, a division of IBM since 2003. RUP is an accumulation of best practices that were collected from thousands of successful projects.

The process itself is configured to adapt to each project specifics. A process specialist, or possibly the project manager, is assigned the task to create, customize, and update the development process.

Figure 2-5 illustrates an overview of the standard Rational Unified Process.

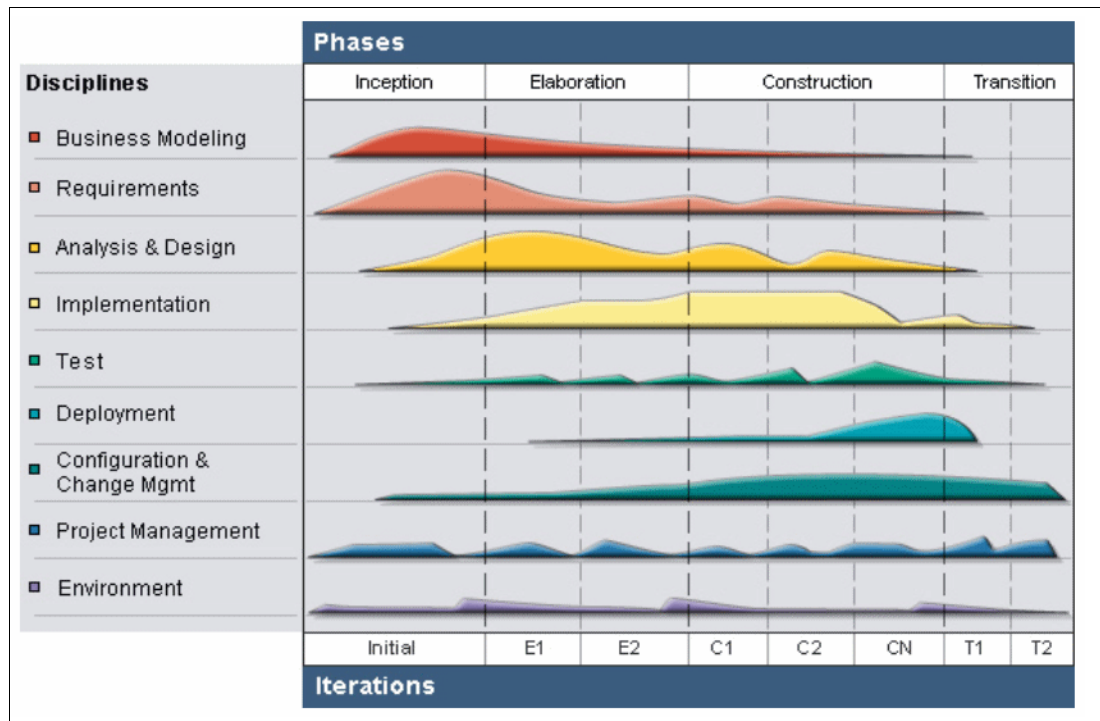


Figure 2-5 The Rational Unified Process overview

The process illustrated in Figure 2-5 is composed of two axis: the vertical axis is labeled Disciplines, and the horizontal axis is labeled Phases. Disciplines are a categorization of tasks that are performed throughout your project life cycle. Phases are the time lines of the project progress and within the phases there are iterations of activity.

In each project phase, you perform most of the activities in the disciplines, but with different intensities and maturity, for example, consider the inception phase, your project team will perform a significant amount of business modeling and requirements gathering. During this

period, to a lesser extent, the team performs some analysis and design, implementation, and so forth.

2.4.2 Process modeling for SOA through SOMA

Service Oriented Modeling and Architecture (SOMA) is composed of three major activities: identification, specification, and realization. It is important to mention that these major activities are tackled in an iterative approach.

Process modeling is central for successful service identifications that results in more agile architectures. Figure 2-6 displays how SOMA links component business modelling and SOA activities together.

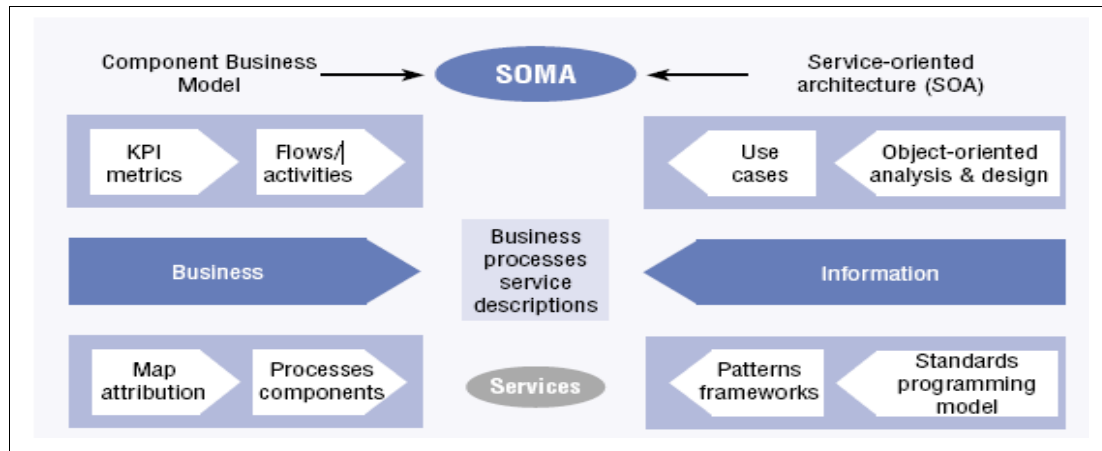


Figure 2-6 Overview of the SOMA environment

Pure IT service identification might not result in robust and agile architectures, which is because the services from the IT perspective might be considerably different from services from the business perspective. Similarly, process models might, or might not, be feasible given the existing infrastructure. To-be process models are sometimes wishful and considerable compromises are undertaken to make them useful. Therefore a combinational approach of service modeling and process modeling are essential to achieve service identification activities that result in robust and agile solutions. Table 2-1 illustrates the major differences between process modeling and service modeling.

Table 2-1 Process and service modeling

Process modeling	Service modeling
Business driven	IT driven
Top-down process approach	Bottom-up architectural approach
Reuse process model	Reuse service implementation
Project oriented	Enterprise infrastructure oriented
Success measured by business metrics and key performance indicators	Success measured by architectural metrics, logical consistency, ease of integration, and cost of change

SOMA was developed by IBM to assist in the alignment of business and IT goals through service modeling that is designed to connect an enterprise's business model with its

technology model. For more information, refer to the IBM Business Consulting Services paper at URL:

<http://www-935.ibm.com/services/us/gbs/bus/pdf/g510-5060-ibm-service-oriented-modeling-arch.pdf>

In the next sections, we present the phases and disciplines of a typical IMS SOA project. It is important to always remember the following points about the process throughout this chapter:

- ▶ Phases are composed of iterations, and each iteration ends with a milestone.
- ▶ Milestones represent a check point in the project. Progress is measured against the work products or the artifacts outlined in each milestone.

2.5 Phases of IMS SOA projects

As in any successful IT project, the initial research, business case creation, implementation, and monitoring of IMS application modernizations follow defined and staged steps. No one process can be applied to all situations, so use these guidelines in a manner that performs best for your particular project.

Also the activities that are outlined in the following phases are not developed strictly in sequence; rather, they are developed iteratively.

2.5.1 Phase 1: Determining if a SOA project is worth considering

SOA focuses on reuse, composite applications, and agility. SOA, is a journey, rather than a single one-time project. SOA projects typically span the enterprise and are not focused within a specific department. It is a corporate-wide initiative that leverages technology in a manner that reflects the business's goals and culture. Both business management and IT must work together to be successful, utilizing working groups that span departments.

Before an IT modernization project is undertaken, you must consider whether your environment is compatible to this effort. There are several questions that assist in this decision. If you have difficulty in answering the questions positively, it might be premature to embark on an IT modernization project until you can provide positive answers.

- ▶ Do you have the skills to both develop and support such projects? If not, can you purchase and retain such skills?
- ▶ Are there documented requirements from your external clients or suppliers that relate to On Demand services using SOA methodologies?
- ▶ Can you quantify the return on investment from such projects, for example, if product time to market is vital, will it be positively influenced?
- ▶ Will these IT modernization projects align with current corporate governance control for security, audit, service level agreement alignment, performance, manageability, and billing?

Here are more questions to consider:

- ▶ The extra 'layering' of support levels makes a difference on overall responsiveness. If increased time for responsiveness is going to be unacceptable, this constitutes a concern. Create and exercise benchmarks early in the project to determine if the increased processing time in a SOA environment is acceptable.

- ▶ Is there a requirement for tightly coupled applications and data? This is normally outside the scope of SOA design.
- ▶ Also, as described in Figure 2-7, is your organization at least at a maturity level of 3 - 'Proactive'?

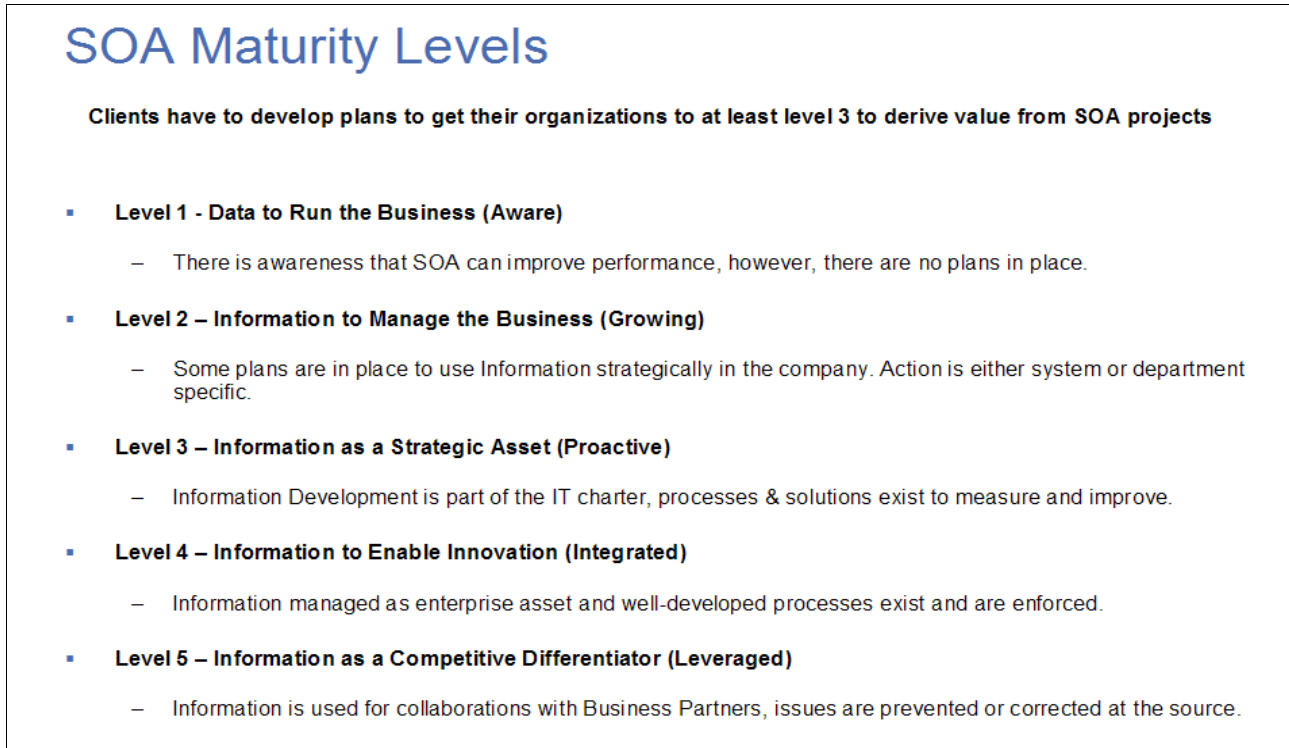


Figure 2-7 Maturity levels of organizations planning for SOA implementations

Outcome of phase 1

Under the assumption that a project can be considered, we now stage through the other suggested phases for successful IMS integration into enterprise SOA.

2.5.2 Phase 2: Developing the SOA project office and plan

IMS modernization to SOA is a corporate-wide undertaking and a comprehensive project plan is required. There are some fundamental choices to make here:

- ▶ Can the project be segmented into discrete components that can run in parallel? For instance, the implementation of the software components, such as activating OTMA, IMS Connect, or IMS TMRA, might occur as Java code is being developed for Enterprise Java Bean use.
- ▶ What will the duration of these project phases be? What external events, such as application development freeze periods, affect the project?
- ▶ What is the representation in your SOA implementation project office and what are their roles?

It is the responsibility of the SOA project office to respond to these questions. There are existing skills that are required, and these staff assignments are probably not new to you because they were utilized in more traditional implementations. The membership of the SOA project office can be made up of:

- ▶ IT project manager: Overall management and leadership responsibility for the project team.
- ▶ Systems administrator: Manages the business and service level agreements.
- ▶ Business analyst: Uses the component business modelling approach to model an enterprise into components to identify opportunities for innovation.
- ▶ Security specialist: Defines security policies and the implementation of them.
- ▶ System and database administrator: Installs and maintains the technical infrastructure.
- ▶ Mainframe service developer and deployer: Obtains the development artifacts and installs them in the target runtime environment.
- ▶ Service integration tester: Responsible for the standard test stages of function, integration, stress, and acceptance.
- ▶ The tool smith: Designs and implements the project specific scripts, generators, and other utilities needed by the SOA implementation team.
- ▶ Knowledge transfer facilitator: Provides access to technical instructors.

But in the SOA world there are additional roles that emerged and must be filled. We refer to the IBM publication *Service-Oriented Architecture Compass: Business Value, Planning and Enterprise Roadmap*: ISBN:9780131870024, which provides general guidance on SOA project planning. It is from this publication that additional skill sets are added to complete the SOA project office personnel, as seen in Table 2-2 was sourced.

Table 2-2 New roles involved with SOA projects

Project role	Performed tasks	Collaborates with	Prerequisite skills	Supporting tools
SOA architect	Solution outline, requirements analysis, architectural decisions, component and operational modeling, communicating business issues and components to services	Any other team member plus line-of-business (LoB) representatives	General IT architectures, Java EE technology, XML, XML schema Web Services and SOA concepts, knowledge of platforms, understanding of best practices, business knowledge	UML editors, office suites
Service modeler	Interface contract design, WSDL editing	Business analyst, architect, service SOA developer	WSDL, XML, and Java EE technologies	WSDL editors, Java-to-WSDL generators
Process flow designer	Business process modeling, assembling atomic services into processes	Service modeler, business analyst, SOA architect, LoB representatives	BPEL and WSDL	Graphical flow modeling tools, corresponding runtime support

Project role	Performed tasks	Collaborates with	Prerequisite skills	Supporting tools
Service developer	Service provider coding, service requestor coding, provide SOAP header handlers if needed, code documentation	SOA architect, Service modeler, and Inter operability tester	Java EE, XML, SOAP, and WSDL	Web services wizards in IDEs, WSDL-to-Java generators
Inter operability tester	WSDL inspection, SOAP envelope tracing, conformance testing, and troubleshooting	Service developers (requestor and provider side)	WSDL, and WS-I profiles	TCP/IP tunnels and monitors, WSI test tools

It is worth noting that one resource can fill in more than one role, or more than one resource can perform the same role. This depends on the size and nature of your project.

It is certainly possible and advantageous to fill the positions with staff that are currently performing somewhat similar functions, for example, current IT architects can, with training, fill the SOA architect role.

Outcome of phase 2

After this phase, a SOA project office receives the important support and authorization from management and executive staff. A business case is drafted.

A project plan is partially completed. Remember, the project plan is continuously updated as you collect more information about the project and reflects any changes throughout the lifetime of the project.

2.5.3 Phase 3: Integrating the IMS modernization project into a governance model

SOA bridges business processes, services, business value, and IT justification. SOA governance is the framework to bring it all together.

Governance provides for a framework to define the activities, actions, authority, and metrics to realize the business and IT benefits of SOA in compliance with legal boundaries. An IBM paper entitled “*Effective SOA governance, March 2006*” states:

“There are two fundamental aspects of governance. The first aspect involves the processes established by an organization to determine who is empowered to make certain decisions. The second aspect includes the mechanisms and policies that are used by the organization to measure and control the way those decisions are implemented. Together, these aspects form a governance framework.”

This white paper is located at:

<ftp://ftp.software.ibm.com/software/rational/web/whitepapers/soagov-mgmt.pdf>

During this phase, you respond to many questions. We used the IBM white paper entitled “*Enabling SOA through organization change and governance, November 2007*” as a source for some of these questions. The URL reference for this white paper is:

http://download.boulder.ibm.com/ibmdl/pub/software/solutions/soa/gov/GBS_SOAgov_Org_Change.pdf

Some representative questions that are associated with this phase are:

- ▶ What are the common business services that are needed?
- ▶ Who owns the services or SOA in general? Who decides on whether a service is accessible to other applications? Who is allowed to change a service that is currently reused by others? Who needs to be responsible for approving a change?
- ▶ How does business and IT align to develop these services collaboratively?
- ▶ Who intermediates between business units that have functional, technological, or quality of service dependencies?
- ▶ Who pays for the initial training or outsourcing skills?
- ▶ Who pays for ongoing maintenance and who is responsible for providing it?
- ▶ Who pays for modifications that are not required by all of the users?
- ▶ Who is responsible for funding changes if there is a need to update infrastructures?
- ▶ Who owns the data that is probably spread across multiple IT platforms?
- ▶ How do you ensure the effectiveness of the implementation metrics? Who defines and owns the service level agreements?
- ▶ What policies and standards are to be common across geographical regulatory jurisdictions, and which are different?

Figure 2-8 outlines the key processes that you must examine to implement SOA solutions.

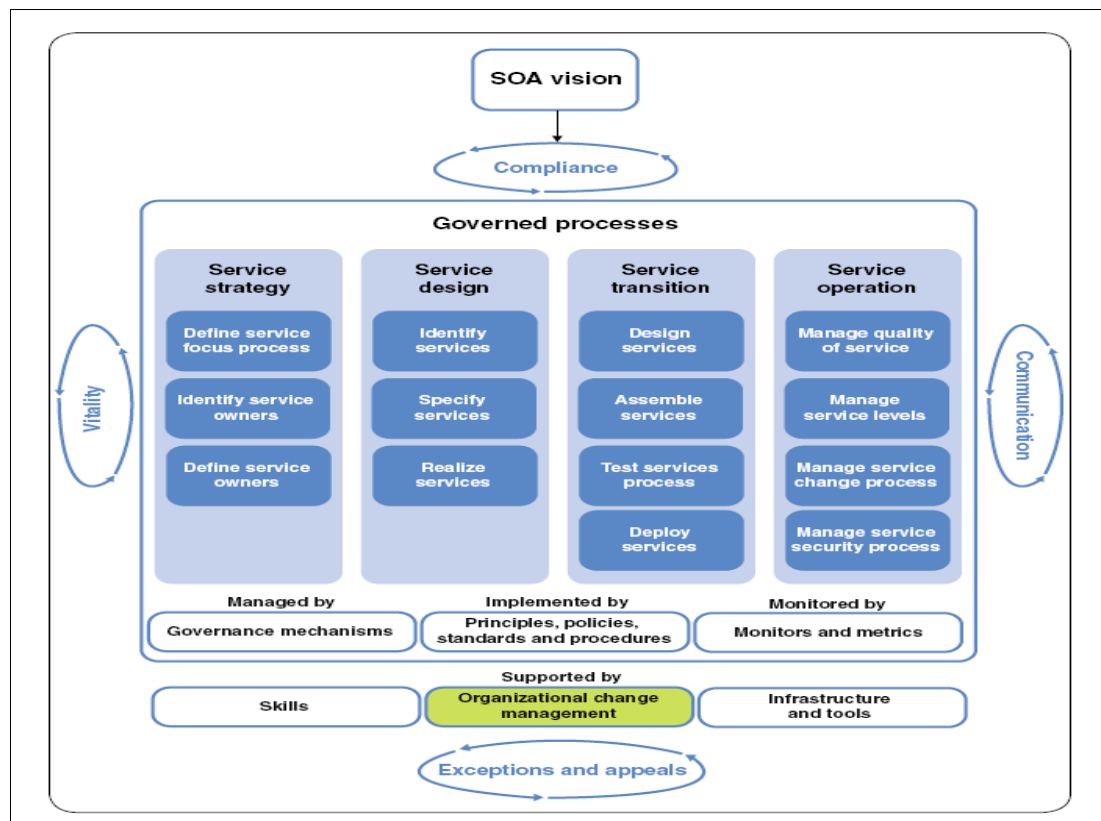


Figure 2-8 Elements of the governance model

To learn more about SOA governance:

Visit the following Web site:

<http://www-306.ibm.com/software/solutions/soa/gov/>

Another excellent resource is:

“Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap”: ISBN-10: 0-13-187002-5; ISBN-13: 978-0-13-187002-4

Outcome of phase 3

Phase three can be performed in parallel with phase two because integrating IMS SOA implementation projects into an enterprise-wide governance model is a considerable undertaking.

The outcome of this phase provides the core material to continue to develop the business case. Also the project plan will have matured during this phase by development of the service implementation strategy, and the design, transition, and operational stages of this project.

2.5.4 Phase 4: Identifying candidate services and flows

Remember that SOA is not only an IT project, more importantly, it is a business project. Introducing SOA has an impact on the business processes across the organization. SOA introduces additional capabilities and automation to the existing business processes. During this phase, you anticipate the change and assess its impact. Service identification, specification, and realization tasks are performed in this phase.

Service identification

Service identification is critical to the success of SOA because it is where IT meets the business. Although service identification leverages existing best practices, such as classic Rational Unified Process (RUP), it is an area where SOA brings innovative concepts and ideas that other paradigms did not address.

This process consists of a combination of top-down, bottom-up, and middle-out techniques of domain decomposition, existing asset analysis, and goal-service modeling.

► Top-down technique:

A blueprint of business use cases provides the specification for business services. This top-down process is often referred to as domain decomposition, which consists of the decomposition of the business domain into its functional areas and subsystems, which includes its flow or process decomposition into processes, sub-processes, and high-level business use cases.

► Bottom-up technique:

Existing systems are analyzed and selected as viable candidates for providing lower cost solutions to the implementation of underlying service functionality that supports the business process.

► Middle-out technique:

This technique consists of goal-service modeling to validate and unearth other services that are not captured by either top-down or bottom-up service identification approaches. It ties services to goals and sub-goals, key performance indicators, and metrics.

The goal of service identification is to create an initial set of candidate services and their associated operations. During service identification, the service model work product is created. At the end of service identification, it is handed off to the software architect(s) who are responsible for service specification. You can think of service identification as producing the analysis level of the service model and service specification the design level.

Tool Support: WebSphere Business Modeler plays a central role in supporting the identification of candidate services and flows. We discuss the WebSphere Business Modeler role in Chapter 3, “SOA Tooling” on page 57.

Service specification

The goal of the service specification activity is to fully specify the elements of the SOA design that are architecturally significant. For this reason, software architects, optionally with the participation of designers, primarily perform service specification.

Service specification is about the A (architecture) of SOA and as such is a key activity without which a SOA effort cannot be successful. Whereas service identification is viewed as the analysis of the service model, service specification is viewed as the design of the service model. During service specification, the service model is fully specified. After service specification, the designers who are responsible for service realization and the developers who are responsible for service implementation use it.

Tool Support: You can use Rational Software Architect for WebSphere Software Version 7.5 to perform service specifications.

Service realization

The service realization activity includes the steps that are required to create a fully-specified design model that is ready for implementation, which includes creating the design and service component models, creating the realization classes, the refining activities, and validation activities.

Service realization is followed by service implementation, assembly, and testing as described in detail in 2.5.9, “Phase 9: Implementing, testing, and monitoring IT services” on page 51.

Modelling the existing business processes (As-Is Model)

To properly identify business and IT services, the first step is to model the existing business processes. This modeling helps you to properly identify the business services that are required to satisfy your new and anticipated business processes.

Business Process: A business process is a set of business-related activities that are invoked in a specific sequence to achieve a business goal. Business processes consist of tasks, which are comprised of:

- ▶ Human interactions
- ▶ Automated workflow
- ▶ Information services
- ▶ Business rule interactions
- ▶ Sub-processes
- ▶ Invocation of functions and services

The difference between a process and a service is that a process is made of tasks that are represented as a separate service or set of services, whereas a service represents an individual, repeatable business task.

Many organizations maintain elaborate models of their business processes. These models document the existing business processes and function as a communication medium within the organization. In addition, business process models help organizations to identify bottle necks and to improvement initiatives.

Modelling the new and improved business processes (To-Be Model)

The To-Be Model is the business process as we expect it to be after the modernization activities are completed. If the modernization project is divided into more than one project or phase, then you might have more than one To-Be Model.

The To-Be Model can be utilized to perform the following:

- ▶ Assess the impact of the change, in terms of effort and duration of performing the tasks.
- ▶ Estimate the anticipated workload on critical IT assets.
- ▶ Identify bottle necks that are introduced in the new process model.
- ▶ Anticipate and plan for reorganization and training.
- ▶ Simulate business processes to:
 - Generate statistical reports on performance and related parameters.
 - Plan for schedule outages, and estimate the number of affected users.
- ▶ To-Be models function as a communication medium between the development team and business analysts.

Ensuring agility and adaptability constraints

SOA promises agility to quickly and cheaply develop composite applications to satisfy ever changing business needs. Agility is hard to measure, but in general we know that loosely coupled architecture as a foundation of SOA has the potential to be more agile.

Architecture agility is closely related to the way business and IT services are defined.

IT service: A building block of SOA and composite applications.

Business service: A building block of the business processes. A business service can be composed of one or more IT services.

Business services are distinctly different from IT services by the fact that business services must have a business value, although the IT service might, or might not, have a business value.

Figure 2-9 on page 43 illustrates a sample business process model and the IT services that realize a subset of the business processes.

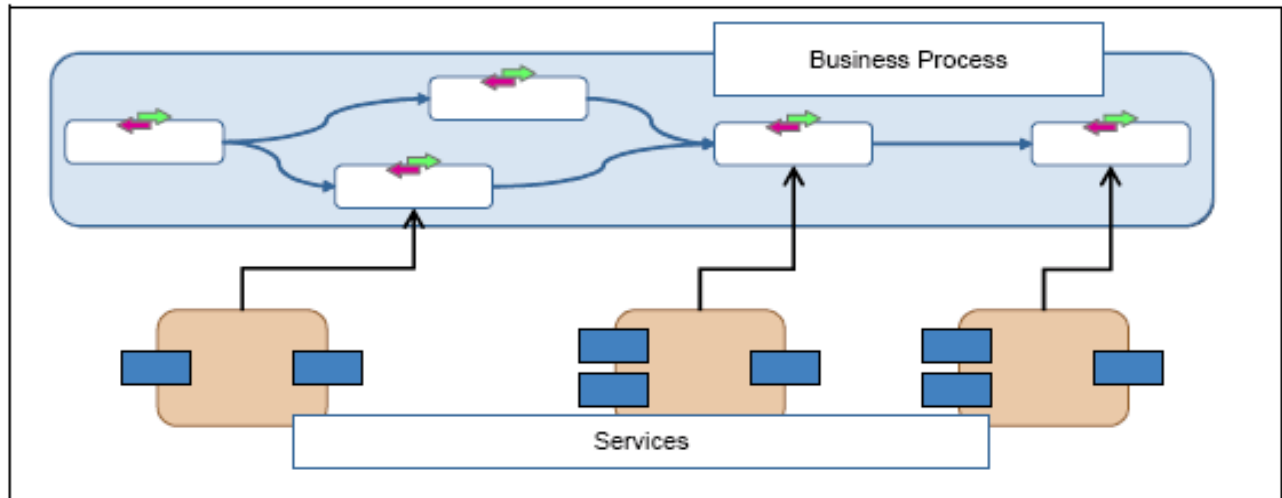


Figure 2-9 Business process flow and IT services that support them

Agility, from this perspective, can be defined as how much do we have to change (edit, delete, create, or reconnect) the IT services in response to a change in the business process model. Our goal is to keep this change as minimal as possible by, for example, reconnecting a number of services rather than editing and creating new services.

Through this audit, you will understand how your application and service code relates and what data stores and external applications are accessed.

- ▶ There are standard methodologies that you can utilize that can identify existing assets and define business components:

- Component Business Modeling (CBM): CBM is a business consultant method by IBM Global Technology Services to decompose an enterprise into its constituent business components. A component is a logical grouping of people, technology, and resources that delivers specific business value, and can operate independently. For more information, refer to this Web site:

http://www-935.ibm.com/services/us/gbs/bus/html/bcs_componentmodeling.html

- Service Oriented Modeling and Architecture (SOMA): The process of service-oriented modeling and architecture consists of three general steps: identification, specification and realization of services, and components and flows (typically choreography of services). For more information, refer to the following Web site:

<http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/>

Although we do not focus this chapter on tooling, Figure 2-10 on page 44 does introduce a few excellent tools that can assist you in discovery and analysis of existing applications, create re-usable logic, and govern services.

IBM Enterprise Modernization Solutions		
Enterprise Modernization	If you need to...	...IBM Software can help
Asset Modernization	<ul style="list-style-type: none"> ✓ Discover and gain control over application assets and their dependencies ✓ Extract business rules currently imbedded in code ✓ Identify key assets and restructure for reuse in SOA ✓ Reduce cost and time of on-going application maintenance ✓ Manage and govern the design, development and consumption of software assets and services ✓ Identify all assets that could be affected by proposed changes 	<p>IBM WebSphere Studio Asset Analyzer – helps IT personnel with the discovery and analysis (DNA) of existing enterprise applications. Understand and gain intellectual control over your application relationships and structures.</p> <p>IBM Rational Transformation Workbench - helps accelerate strategic and tactical modernization initiatives by allowing development team to quickly transform existing assets and discover reusable business logic for creating services.</p> <p>IBM Rational Asset Manager – helps improve productivity and software delivery through asset reuse with a solution that enables you to create, modify, govern, and locate any type of development assets, including SOA and systems development assets.</p> <p>IBM WebSphere Service Registry and Repository – helps you achieve tangible business value from your service-oriented architecture (SOA) by enabling better management and governance of your services.</p>

Figure 2-10 Suggestions for tools to utilize during the development process

Outcome of phase 4

An identification of the existing data and application models that are in existence.

2.5.5 Phase 5: Identify your specific IMS assets

In this section, we provide suggestions on the work items that are associated with this project phase.

Identify and analyze your existing technologies supporting your core business services. SOA services for IMS can be defined under three categories:

► Reuse existing transactions

Reusable existing transactions and programs must have the following characteristics:

- Start with existing mainframe technologies and applications that can act as services.
- Select a business process that spans two systems at most.
- Use only basic services and leave complex service composition for later when more experience is available in your organization. Examples of more complex services are applications that use conversation processing.
- Implement a simple, single message exchange pattern (request / response).
- An input / output message format must be available from the following sources:
 - Cobol copyBook, C includes, PL/I includes. If it is not available, it must be easy to build from information in the application program
- Avoid applications that use programmed attributes for Message Formatting Services (MFS).

► Reuse pieces (callable modules) of existing transactions to build new interfaces:

- You must build a new message interface for the view component.

- From the view component, the existing code is called. Several possibilities are available:
 - Using DB2 stored procedures has the advantage that we have language isolation (for example, the view and controller can be in Java and the existing modules in Cobol).
 - Programs that are written in Java composed of classes and by definition, modular.
 - Traditional coding techniques available in Cobol, and PL/I.
- ▶ Build new transactions

New application programs can be built with 3GL languages (Cobol, C, PL/I, ASM) and Java. The Java language gives the opportunity to access the DLI and DB2 data through JDBC.

The new programs have the following characteristics:

- Well designed message interface
- Non-conversational
- Modular built-ups
- No special multi-segment requirements
- Consider the usage of a 4GL, such as Enterprise Generation Language (EGL) that is available with the Rational Business Developer Extension. Rational Business Developer is an Integrated development environment (IDE) developed by the IBM Rational Software division that provides Eclipse based tools for development of cross platform applications and services using the EGL Language (a business oriented programming language). For more information, refer to:

<http://www.ibm.com/developerworks/rational/library/may07/sergi/index.html>

To elaborate on the identification and analysis of existing transactions during this determination phase of IMS assets to consider, we use diagrams that Suzie Wendler of the IBM Advanced Technical Support (ATS), Americas created.

Figure 2-11 on page 46 presents the simplest models of operation that existing applications can perform.

The first model is associated with a GU to the IOPCB to pull an input message off the queue and a direct ISRT call to place the reply onto the message queue, and eventually the inputting terminal or application.

The second model relates to a message switch where the first application ‘spawns’ another, and that application replies to the inputting terminal or application.

- How do your traditional interactions between programs translate to new interoperability modes?

- Application models:

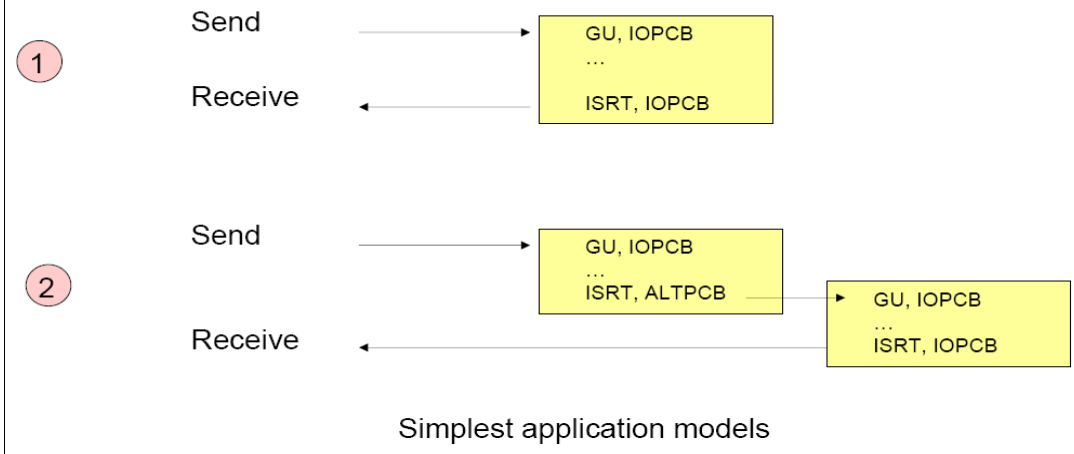


Figure 2-11 First two of eight potential application models in use

Figure 2-12 on page 47 presents more application models:

- ▶ Model 3 relates to multi-segment output message flows or multiple messages that are delimited through PURGE calls to force single multi-segment message flows.
- ▶ Model 4 represents multiple IMS transactions where the spawning transaction always replies before the spawned transaction.
- ▶ Model 5 demonstrates a single IMS transaction spawning multiple transactions. Any of those transactions can reply first, based on whichever one issues a GU to the IOPCB first. There there is no enforced reply serialization in this model.

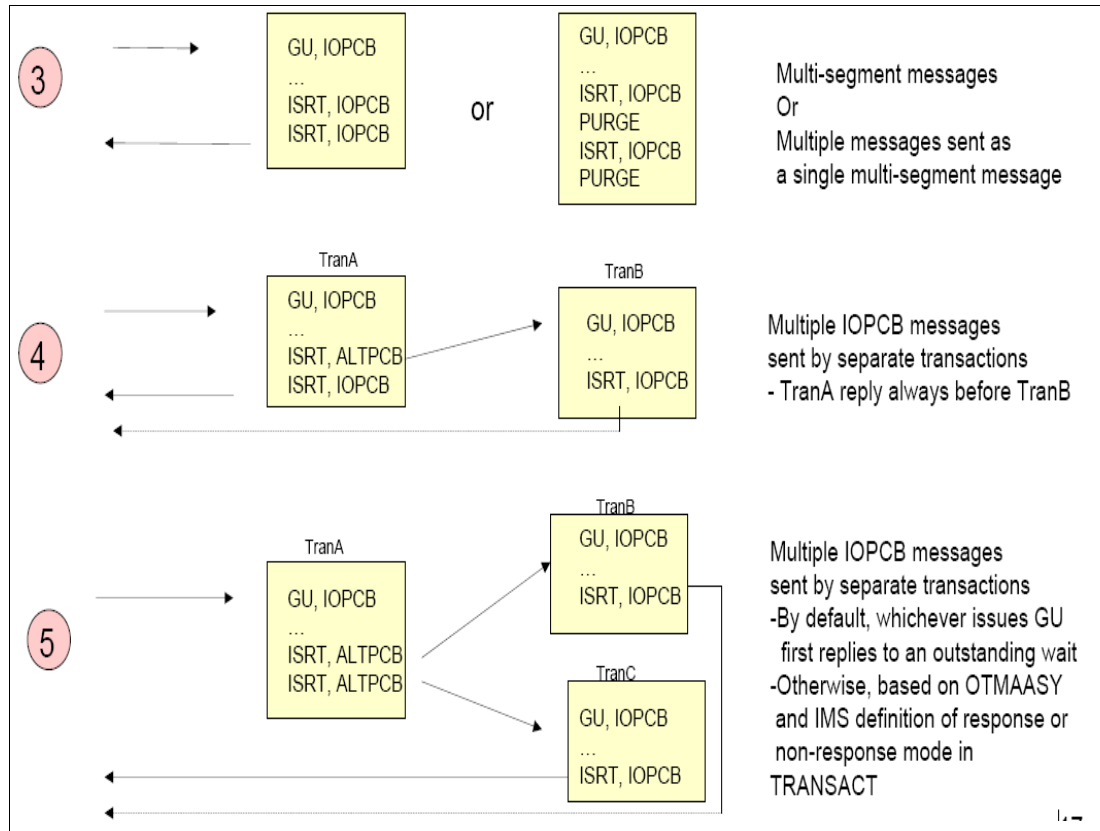


Figure 2-12 Other potential application models in use

Figure 2-13 on page 48 presents more complex application models that are in use in some existing Client environments:

- ▶ Model 6 presents the case when the remote program is defined as CM1 (send-then-commit) and as such waits in synchronous mode to complete the commit cycle. A spawned application does not reply, and a DFS2082 message "A response mode application terminated normally without having an output reply message returned to the terminal" is sent to reset response mode after the application terminates.
- ▶ With Model 7, both messages that were targeted by the IOPCB and ALTPCB are sent to the same destination. In traditional IMS environments, the PA1 key is used to obtain subsequent segments as a page advance and this application call flow provides for similar support.
- ▶ Within Model 8 the target application does not reply to the original input and a DFS2082 is issued but a message also inserted through the ALTPCB to the inputting program or terminal.

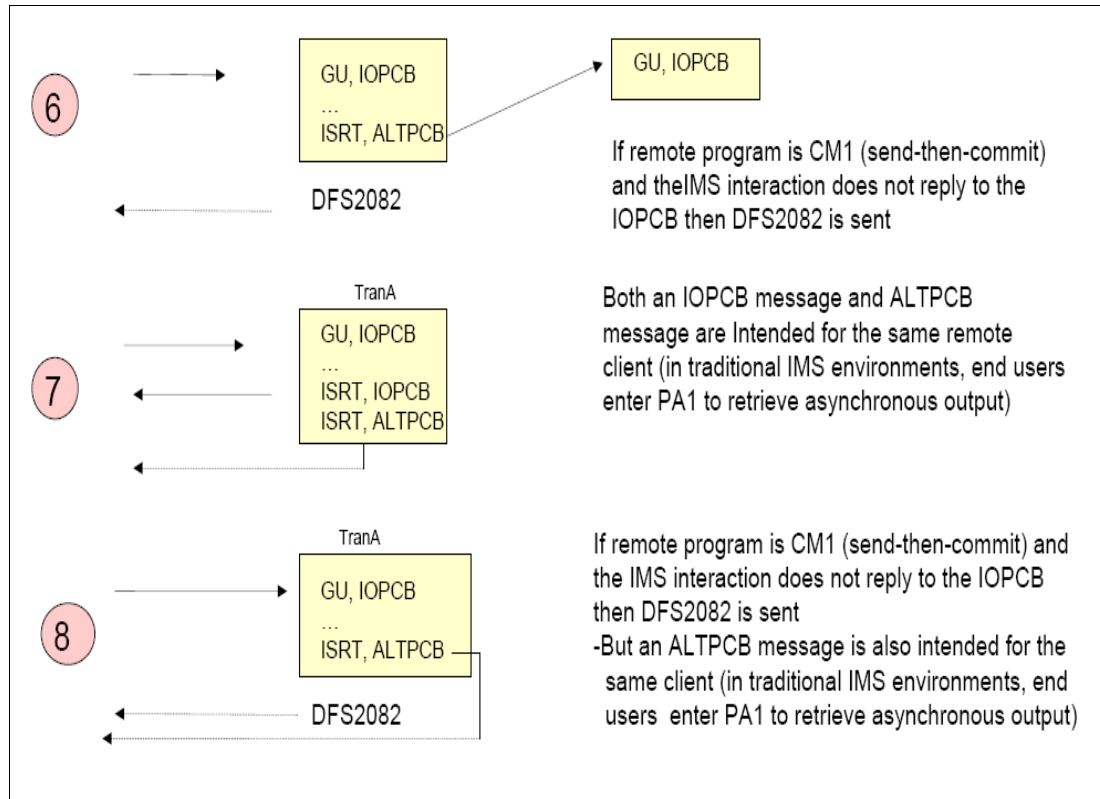


Figure 2-13 Other potential application models in use

As you can see, when selected applications are chosen to receive a distributed front end interface we must understand original IMS application designs to create robust solutions, for example:

- ▶ You must consider definitions, such as CM0 (Commit-then-send), which is more asynchronous in nature, and CM1 (Send-then-commit), which is a more synchronous interaction.
- ▶ You must design for the submission of messages that are produced from calls using IOPCBs or ALTPCBs from one input.
- ▶ There must be support for understanding when all messages are received and how to correlate unsolicited messages.
- ▶ Do your applications take advantage of MFS capabilities, such as adding data into messages? If so, the remote application design might require an understanding of the coding requirements to handle additional message data in the OTMA, MQ, and IMS Connect headers.

Outcome of phase 5

Expect to identify those applications, their accessed data stores, and external systems that are acceptable candidates for IMS modernization.

2.5.6 Phase 6: Evaluating architectural changes and technologies to utilize

It is obvious that an IMS modernization project results in the use of new technologies. The entire set of chapter headings in Part 2, "Creating, deploying, and managing IMS services" on page 69 introduces you to these.

We offer an example of deploying IMS services in Chapter 4, “Example of deploying an IMS Service” on page 71 to supply you with the rationale and implementation flow to resolve a sample business modernization project.

Some questions that you address during this phase are:

- ▶ Will architectural changes be made? If we must convert from COBOL or PL/I to object oriented Java, it is a more difficult endeavour compared to leaving the heritage programming alone. We assume that mid-tier hardware and software is required. If this is not currently in place, it must be factored into the business case.
- ▶ What technological choices will be made? IMS can be both a service requestor and provider and can be linked to Web-based components through a variety of options, such as SOAP Gateway, IMS TMRA, Open Database, WebSphere, and so on. A major goal of this book is to explain the options that are available and to provide guidance on when to choose them.
- ▶ Does this architectural environment offer sufficient performance to present the end user with similar response characteristics experienced before hand?
- ▶ What level of training is required to understand, implement, and maintain these new architectures and technologies?

There are several modernization approaches that can be followed with SOA but non-invasive reuse is best aligned with IMS. The existing applications are not modified or the modifications are minor and follow documented APIs, which is the case in using the synchronous callout DL/I ICAL call. Web-based services are layered on top of the existing IMS environment, which results in the least amount of development cost.

Outcome of phase 6

Expect to map the architectural and technical components that construct your IMS SOA project.

2.5.7 Phase 7: Creating an Enterprise Service Bus

An Enterprise Service Bus (ESB) powers SOA by reducing the number, size, and complexity of interfaces. Figure 2-14 on page 50 helps to define what an ESB is designed for.

What is an Enterprise Service Bus (ESB)?

Flexible connectivity infrastructure for integrating applications and services to power your SOA

- ▶ **ROUTING** messages between services
- ▶ **CONVERTING** transport protocols between requestor and service
- ▶ **TRANSFORMING** message format between requestor and service
- ▶ **HANDLING** business events from disparate sources

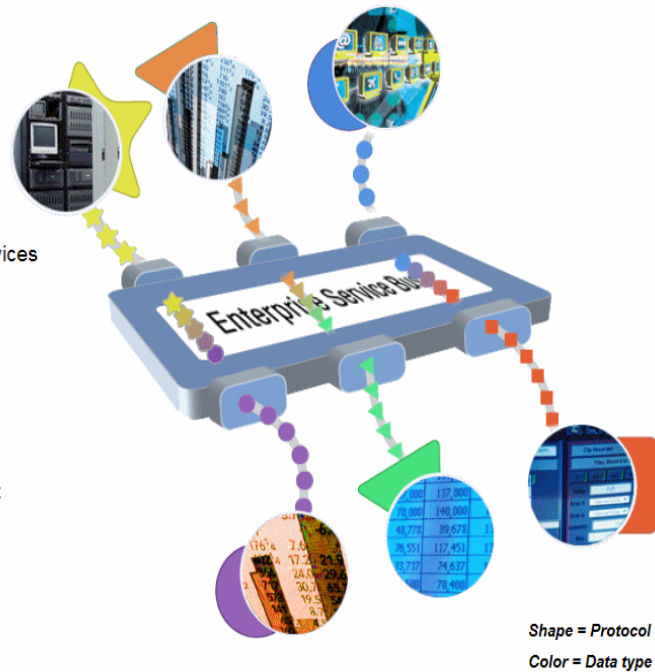


Figure 2-14 Defining an ESB

Outcome of phase 7

Most likely an ESB is selected for purchase rather than created. One such ESB support is available in the WebSphere Message Broker. It delivers an advanced ESB to provide:

- ▶ Universal connectivity including SOAP, XML, JMS, COBOL copybook, SCADA, and so on
- ▶ Advanced message transformation, enrichment, and routing

For more information about WebSphere Message Broker, refer to:

<http://www-01.ibm.com/software/integration/wbimessagebroker/>

2.5.8 Phase 8: Selecting a pilot project

With the additional product and solution sets that are available in the SOA space it is important to choose a proof-of-concept project to represent the code and business processes and present results that can be measured.

During your first SOA integration project it is important to contain the work effort. To succeed, use the criteria listed in 2.5.5, "Phase 5: Identify your specific IMS assets" on page 44 to make the selection of this first pilot project.

Outcome of phase 8

There are a few results that you must expect to obtain from this activity:

- ▶ The project plan is exercised and any opportunity improvements made clear.
- ▶ The tool set is exercised, and you can determine if they provide the advertised functionality.

- ▶ If there are skill set deficiencies, they are observed here.
- ▶ The test and validation strategies that the SOA project office developed are exercised.
- ▶ The most important result is a working SOA implementation that you can demonstrate to decision makers to achieve the required support and funding for further modernization projects.

2.5.9 Phase 9: Implementing, testing, and monitoring IT services

This phase represents the bulk of the development effort because it is here that the identified IT services are implemented. Notice that because the architecture is loosely coupled, those IT services can be developed in parallel.

In this phase, we address the following activities:

- ▶ Implementation
- ▶ Testing
- ▶ Deploying and monitoring

In the following chapters in this publication, and other IBM Redbooks mentioned in Chapter 1, “SOA and IMS: A powerful business combination” on page 1, details about how to select the proper IMS SOA solution and how to implement are presented. In this section, we provide an overview of the major activities in this phase.

Implementation activities

One benefit from loosely coupled architectures is that component and service implementation can be executed in parallel. Figure 2-15 on page 52 illustrates a typical implementation workflow for a service-oriented project. Note that the Rational Unified Process advocates *iterative development*. In iterative development, components and services are developed, tested, and integrated over a number of iterations. The main benefit of iterative development is early identification of risks and shortening of development schedules.

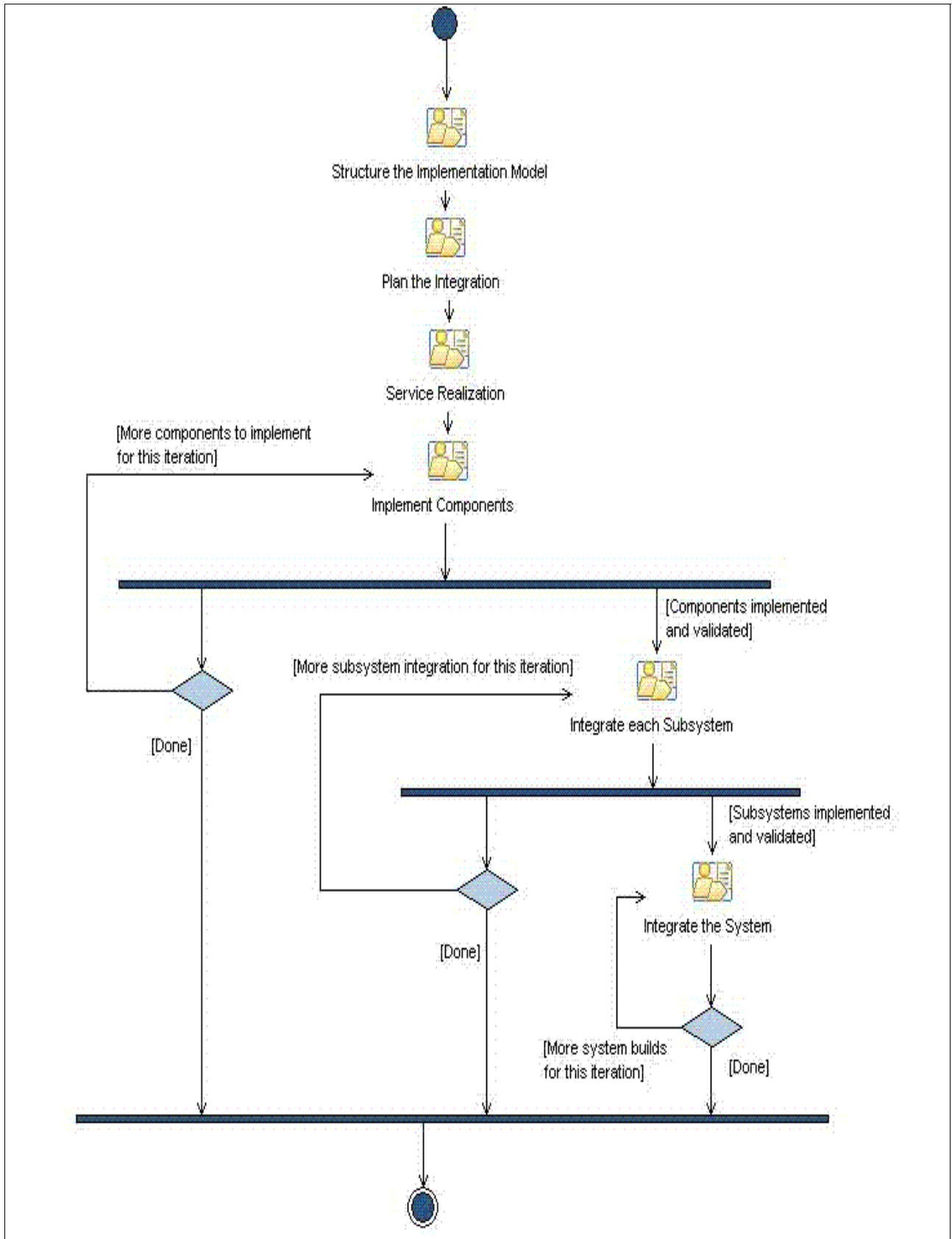


Figure 2-15 Implementation workflow

An implementation example of IMS SOA components are presented in Chapter 4, “Example of deploying an IMS Service” on page 71.

Testing activities

Another benefit of loosely coupled services is that testing can start early in the development life cycle. Refer to our development methodology in Figure 2-5 on page 33. Notice that testing starts early in the development life cycle and provides the following benefits:

- ▶ Better utilization of human resources because testing and development activities are more uniform over the project life cycle
- ▶ Early identification of risks
- ▶ Shortening of the development schedule

Deploying and monitoring

Enterprise services are typically deployed on the Enterprise Component Layer, which is comprised of the enterprise application, process, and workflow servers. Service orientation can result in the introduction of new middleware components to address additional processing or functionality required.

An example of such an additional middle layer component is IBM DataPower. A significant portion of the SOA standards are XML based, which brings about additional data processing and security requirements. IBM DataPower is positioned to meet the additional security requirement and to move data processing cycles from the mainframe to the mid-tier layer. For more information about IBM DataPower, refer to Chapter 16, “Using DataPower with IMS” on page 341.

Modernizing IMS infrastructure systems and integrating its functionality in an SOA environment brings about more emphasis on monitoring requirements. Traditional delays in transaction execution was typically anticipated to a single components, such as IMS. In an SOA environment, performance might have a larger number of dependencies. In addition to the traditional monitoring requirements of IMS transactions, monitoring must include:

- ▶ Network monitoring

Loosely coupled components typically interact by exchanging messages. Loosely coupled architectures inherently create increasing network dependency. Your organization can enforce some service levels if the network is internal, but cases where the network is external to your organization, the challenge is greater.

- ▶ Process performance monitoring

Traditionally, performance monitoring was focused on individual applications or components. As the enterprise solutions evolve towards a service orientation, the scope of performance monitoring inevitably widens.

Let us take, for example, a business scenario, where a client accesses his account balance through a Web-based interface. The request is typically an HTTPS flow through a public TCP/IP network. There might be another layer of XML appliances that handle security and data conversions. The request can be routed to a middle layer application server before being routed to IMS through IMS TMRA, for example. If the end user experiences degraded performance, host system administrators might not have an answer for the issue because IMS performs a smaller part of the process than traditionally. It is important to utilize holistic, real time process monitoring to assist in pinpointing a problem down to a specific computational component of your system.

IBM supplies powerful facilities to monitor your SOA environment. Figure 2-16 on page 54 presents an overview of the types of activities that can be performed through the use of the WebSphere Business Monitor.

WebSphere Business Monitor

Understand, monitor, and explore the state of business operations

Know the state of your business via

Scorecards

Key Performance Indicators for business units

Take action via
Collaboration

Work with teams to resolve situations

Take more action via
Business Alerts

Notification of situations that require response

Evaluate data via
Reports & Analyses

Understanding trends by combining real-time performance and historical information

Have any available information via
External Information

Monitor can access external information affecting performance

See all of this in
one place!



Figure 2-16 Overview of the facilities that are associated with the WebSphere Business Monitor

For more information about WebSphere Business Monitor, refer to:

<http://www-01.ibm.com/software/integration/wbimonitor/>

Process monitoring provides this macro-level performance monitoring. It involves monitoring the performance of the entire business process rather than a small segment of the process. From its name, this type of monitoring is business oriented. Typically, this type of monitoring is based on *Key Performance Indicators (KPIs)*.

Key Performance Indicators (KPIs): KPIs are performance measurements that indicate the overall performance of a business activity or process. Some examples of KPIs are:

- ▶ Number of bank account transactions executed per day.
- ▶ Number of new accounts opened per week.
- ▶ Number of days on average to open a customer account.

These KPIs are not IT centric, but rather are business centric.

In Chapter 3, "SOA Tooling" on page 57, we introduce some of the tooling support for monitoring. Also refer to Chapter 7, "Performance considerations for IMS Services" on page 117 for additional detail.

Outcome of phase 9

A completed IMS SOA service implementation must be the result of this phase.

Although the nine phases that we presented might not encompass all of the stages that are necessary for a SOA implementation, they certainly give you guidance on the major activities that are involved.

SOA Health Check

If you are just starting with SOA or embarked on the SOA journey for some time, you probably asked yourself these questions:

- ▶ Did we get it right?
- ▶ What about security?
- ▶ Can we be sure of the highest level of reliability?
- ▶ Can we support the projected volumes?
- ▶ How quickly can we grow the user base?
- ▶ Are the right processes in place to support the new environment?
- ▶ Is the business getting the promised values?

IBM offers the following Infrastructure Health Check Services for SOA:

- ▶ The IBM SOA Applications and Services Health Check Workshop examines application reuse to verify that existing applications are properly used.
- ▶ The IBM Infrastructure Health Check Workshop for SOA assesses the ability of servers, storage, networking, middleware, and systems management to adapt to spikes in demand.
- ▶ The IBM Infrastructure Architecture Health Check for SOA examines architecture implementations and recommends improvements for optimal performance.

SOA Health Checks: For more information about SOA Health Checks, visit:

<http://www.ibm.com/software/solutions/soa/healthcheck.html>

The next chapter of this book provides more details about tooling for SOA and IMS solutions, and how IMS plays a constructive role in providing and consuming service components in modern service-oriented environments.



SOA Tooling

In this chapter, we introduce a number of SOA-related tooling with emphasis on explaining how each tool plays a role to leverage your existing IMS assets and enables you to have a successful IMS SOA implementation.

The specific objectives for this chapter are:

- ▶ Understand the major tools related to an IMS SOA implementation.
- ▶ Understand the main functionality of these tools.
- ▶ Help you to plan and make decisions that are related to skills reuse and resource planning that is related to SOA tooling.

3.1 Overview of tools support

Although the choice of tools and methodology can change from one implementation to the other, there are a number of central tools that are most probably included in an IMS SOA implementation, which we title IMS SOA core tools.

The chapter is organized in two sections:

- ▶ “IMS SOA core tools” on page 58 covers the tools closely related to IMS SOA implementation. These tools are: Rational Application Developer, Rational Developer for System z, and WebSphere Application Server.
- ▶ “IMS SOA supporting tools” on page 63 covers a number of tools that you can utilize in an IMS SOA project.

3.2 IMS SOA core tools

In this section, we present the core set of IMS SOA development tools, starting with the IBM Rational Developer.

3.2.1 Rational Application Developer

IBM Rational Application Developer for WebSphere Software V7.0 is an Integrated Development Environment (IDE) platform for building Java Platform Standard Edition (Java SE) and Java Platform Enterprise Edition (Java EE) applications with a focus on applications to be deployed to IBM WebSphere Application Server and IBM WebSphere Portal. Using it, Java developers can rapidly design, develop, assemble, test, profile, and deploy high-quality Java/Java EE, Portal, Web, Web services, and SOA applications.

Wizards in Rational Application Developer V7 offer an easy and quick way to build the connector code and other client artifacts. Rational Application Developer V7 also gives you the possibility to extend the connector proxy as an Stateless Session Bean (SLSB) or as a Web Service. If there is a requirement for building an importable service for the WebSphere Process Service or WebSphere Enterprise Service Bus Server, an extension of Rational Application Developer, WebSphere Integration Developer must be used.

For more information, access the Web site:

http://www.ibm.com/software/awdtools/developer/application/?S_TACT=105AGX15&S_CMP=LP

3.2.2 Rational Developer for System z

The IBM Rational Developer for System z consists of a common workbench and an integrated set of tools that support end-to-end, model-based development, run-time testing, and rapid deployment of simple and complex applications. It offers an integrated development environment with advanced, easy-to-use tools and features to help WebSphere, CICS, and IMS developers rapidly design, code, and deploy complex applications.

Rational Developer for System z provides the following capabilities:

- ▶ Web development

Rational Developer for System z helps develop creative and data-rich Web-based applications. It provides visual layout tools to help you write Java Server Faces (JSF), JavaServer™ Pages (JSP™), and HTML. It includes a wizard to develop servlet code and using it you can create Web applications from database queries and beans. Rational Developer for System z includes graphic-design software for producing static and animated output and supports Java EE coding and deployment.

- ▶ Java development

In addition to the creation of Java code, Rational Developer for System z enables incremental compilation and provides common local and remote debugger benefits.

- ▶ Java EE platform support

Rational Developer for System z features full Enterprise JavaBeans (EJB™) support, Web Archive (WAR) and Enterprise Application Archive (EAR) deployment support. It includes an updated EJB test client and an enhanced unit-test environment to create multiple projects with different unit-test configurations.

- ▶ Analysis and performance profiling

Rational Developer for System z can collect and present Java runtime data in graphical and non-graphical views that help detect application performance issues early in the development cycle and improve overall system efficiency.

- ▶ z/OS and UNIX® System Services development

Rational Developer for System z provides an interactive workstation-based environment to help create, maintain, and reuse applications for traditional processing or for inclusion in a SOA. It provides easy access to IBM z/OS and UNIX System Services data sets and creates Assembler, COBOL, PL/I, C, and C++ applications through remote syntax check, content assist, visual BMS mapping and JCL generation capabilities, visual MFS editor, and color coded editing.

- ▶ Rational Developer for System z has the following additional capabilities:

- Debug and test CICS and IMS system-based code.
- Remote compile generation, build, and deployment support.
- Integration with z/OS IBM problem determination tools, for example, Fault Analyzer for z/OS client for Rational Developer for System z V7.1. Fault Analyzer Integration allows users to browse Fault Analyzer ABEND reports on CICS, IMS, Batch, Java, WebSphere, and other runtimes, and view dump selections relating to ABENDs.
- File Manager Integration enables access to Keyed Sequence Data Set (KSDS) files from the IBM Rational Developer for System z workbench.

We now discuss the Rational Developer for System z workbench.

3.2.3 Understanding the Rational Developer for System z workbench environment

Rational Developer for System z is an Eclipse-based tool. If you used an Eclipse-based tool before, then you are halfway in understanding the Rational Developer for System z workbench environment. If not, you can benefit from the workbench introduction in this section.

Figure 3-1 illustrates the Rational Developer for System z workbench Welcome window.

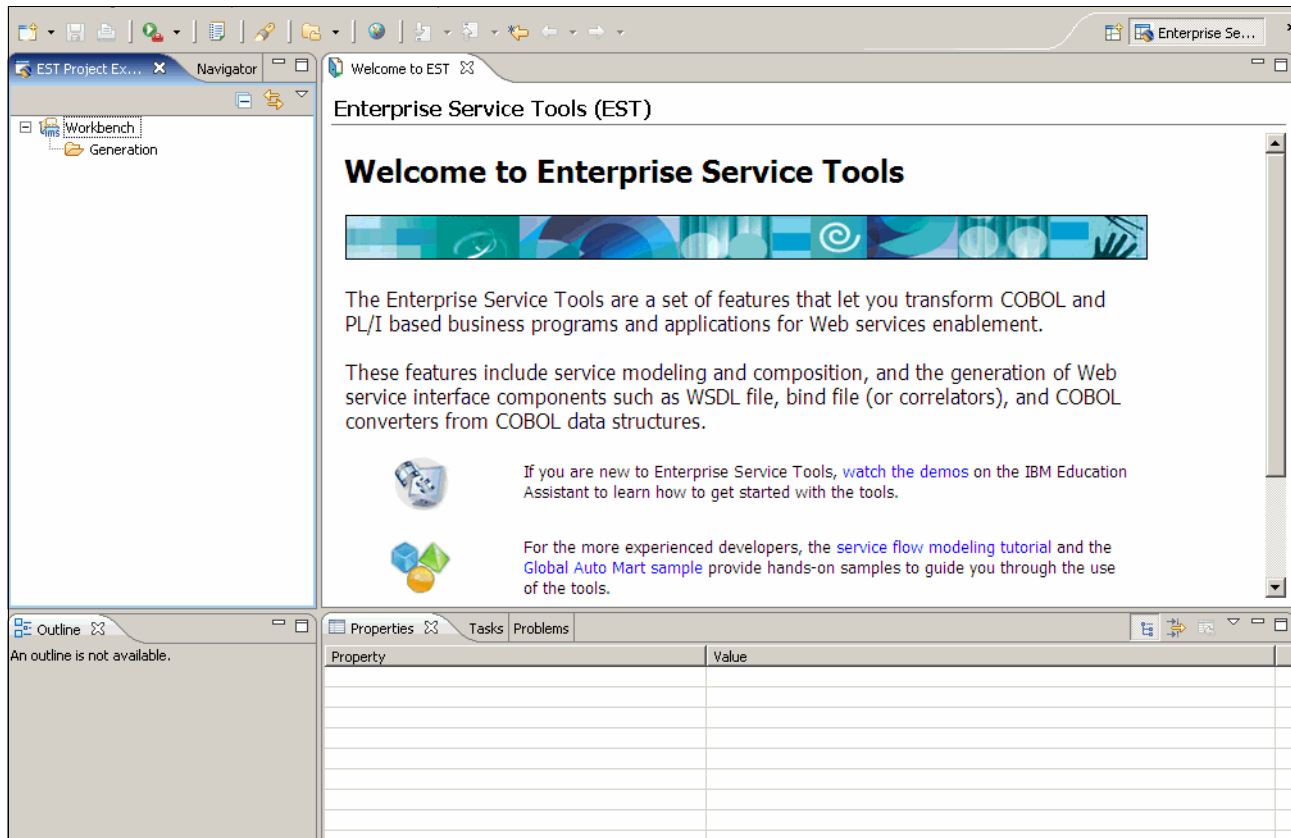


Figure 3-1 Rational Developer for System z workbench welcome window

The main workbench components are:

- ▶ Workbench perspective

A perspective is an arrangement of views. Each perspective contains the views that you need to perform a certain task. You can use the default perspectives that are included with the workbench, or you can create customized perspectives to suit your needs.

- ▶ Project Explorer

The workbench can function as a self-contained file system, from which you can create, edit, share, and delete many different types of files.

- ▶ Views

Perspectives are composed of a number of views that are arranged within your workbench. You can add and remove views from your perspective to customize it to your needs.

- ▶ Capabilities

This is another important feature of the Eclipse-based workbench. Capabilities are large areas of functionality within the workbench. You enable or disable each capability based on your user role, and the workbench uses this information to show or hide options, such as new types of projects and files to create. If you are following a tutorial and cannot find a menu item or a certain functionality, checking your workbench capability is a recommended first step.

To view or update your workbench capabilities:

1. Click **Windows@** → **Preferences**.
2. Click the **General** Tab, and click **Capabilities**. The Capability view shown in Figure 3-2 is displayed.

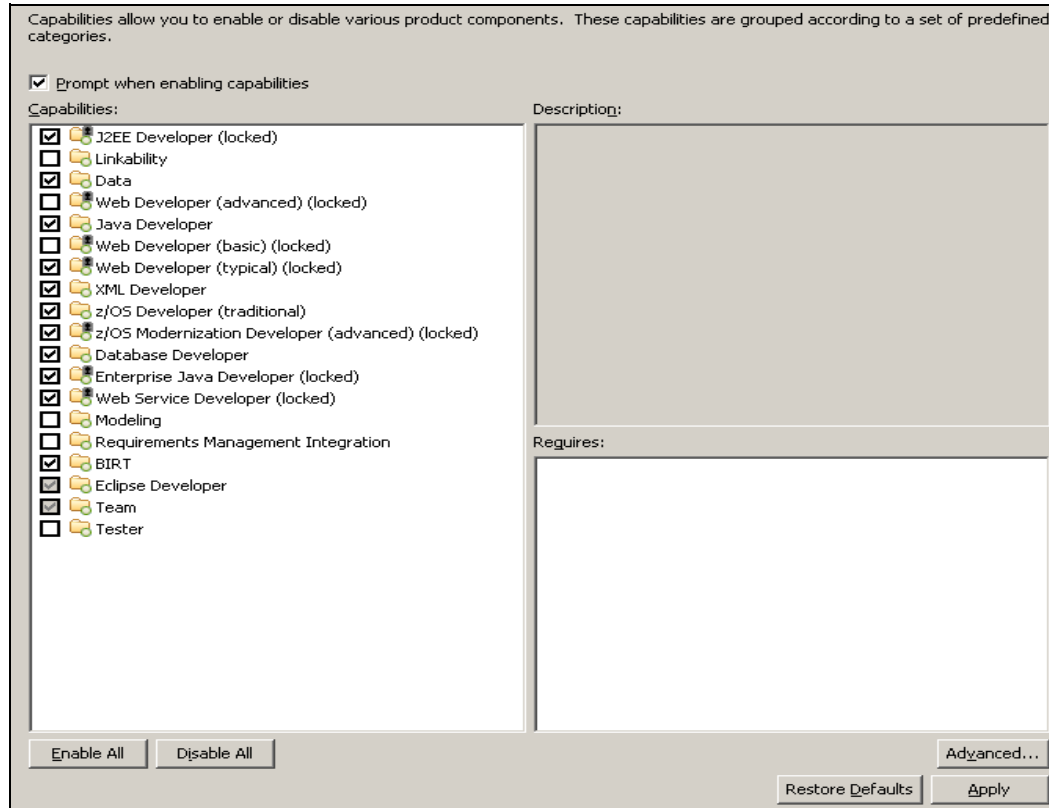


Figure 3-2 Rational Developer for System z capabilities view

Rational Developer for System z:

For more information about the basic Rational Developer for System z interfaces and functionality, refer to:

<http://www.ibm.com/software/awdtools/rdz/>

3.2.4 WebSphere Application Server

WebSphere Application Server is IBM application middleware solution. WebSphere Application Server is built to better deliver the secure, scalable, and resilient application infrastructure that is needed for SOA.

The WebSphere Application Server Family

The WebSphere Application Server family, as depicted in Figure 3-3 on page 62, offers a range of configuration options and supports multiple business models and deployment platforms. In this section, we provide an overview of each WebSphere Application Server configuration to help you choose the appropriate option for your organization.

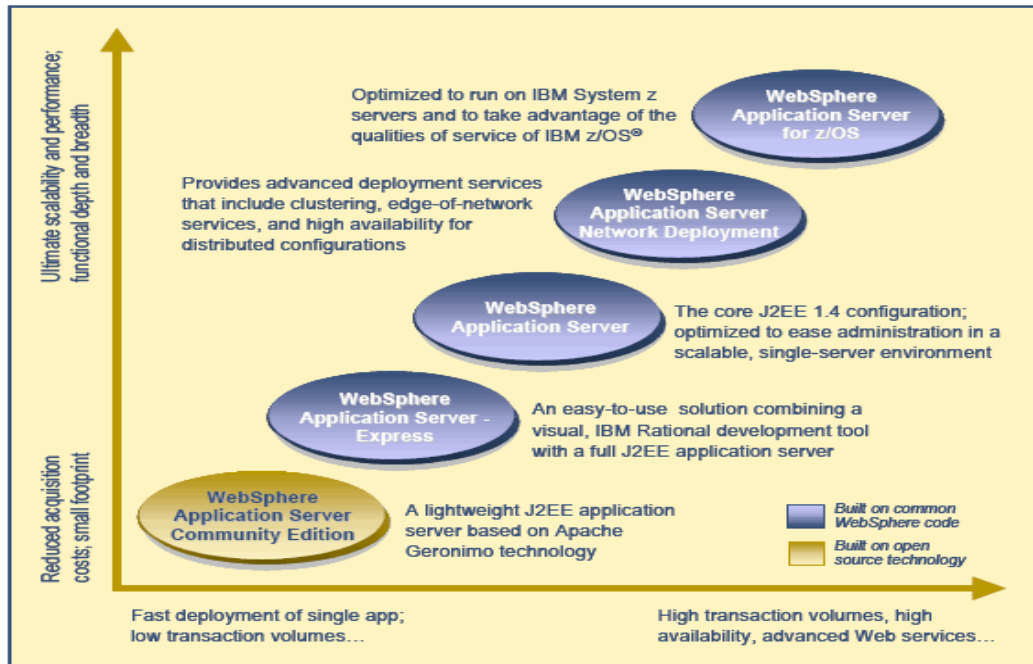


Figure 3-3 WebSphere Application Server configuration options

WebSphere Application Server Community Edition

WebSphere Application Server Community Edition V.11 is an entry-level offering that is built on open source technology from the Apache Software Foundation. The product's intent is to bring the innovation and cost-saving benefits of open source community development and distribution to the WebSphere family, which gives Java developers a highly accessible and flexible platform for building and deploying Java applications.

For more information, access the following Web site:

<http://www-01.ibm.com/software/webservers/appserv/community/>

WebSphere Application Server – Express

WebSphere Application Server – Express is an easy and affordable entry point that combines a visual development tool from IBM Rational with a fully Java EE-compatible application server. A set of integrated applications, wizards, and samples help users to get up and running quickly, and the tight integration between the development environment and the application server enables simpler and faster creation of dynamic Web sites.

For more information, visit the following Web site:

<http://www.ibm.com/software/webservers/appserv/express/>

WebSphere Application Server

WebSphere Application Server, the core offering of the family, is an application platform optimized to ease administration in a scalable, single-server deployment environment. This core Java EE configuration delivers a secure, high-performance transaction engine that is well suited for standalone, departmental applications, and Web services.

For more information, access the Web site:

http://www.ibm.com/software/webservers/appserv/was/index.html?S_TACT=103BGW01&S_CM P=campaig

WebSphere Application Server Network Deployment

WebSphere Application Server Network Deployment is the IBM primary application server offering for businesses and partners who need a full Java EE application platform with higher qualities of service built in. This configuration adds advanced deployment services to the core WebSphere Application Server functionality, which includes clustering, edge-of-network services, and high availability for distributed configurations.

For more information, access this Web site:

http://www.ibm.com/software/webservers/appserv/was/network/index.html?S_TACT=105AD02W&S_CMP=campaign

WebSphere Application Server for z/OS

The z/OS version of the WebSphere Application Server delivers a Java EE application server that is specifically optimized to utilize the unique qualities of service provided by IBM hardware and the z/OS operating system. The programming and deployment models of WebSphere Application Server for z/OS are functionally equivalent to those of WebSphere Application Server Network Deployment, which provides for greater flexibility in development and improved deployment and management functions. WebSphere Application Server for z/OS also provides optimized transactional integration with CICS, IMS, DB2, and VSAM. For more information, access the Web site:

http://www.ibm.com/software/webservers/appserv/zos_os390/

WebSphere Application Server:

There is more information about WebSphere Application Server that is available:

- ▶ Training resources: <http://www.ibm.com/software/websphere/education/>
- ▶ For demos and videos:
http://www.ibm.com/software/info/television/index.jsp?lang=en_us&cat=websphere&item=xml/U939990E82018I09.xml&S_CMP=rnav

3.3 IMS SOA supporting tools

In this section, we introduce some of the tools that are loosely related to IMS but can play a part in SOA IMS implementations.

Successful SOA implementations involve business transformations. It is therefore crucial to manage and plan for such business transformations and understand the impact on the running business processes and performance implications. This brings us to the close relationship between SOA and Business Process Management (BPM).

It is outside the scope of this publication to detail the methodology of BPM when combined with SOA implementation. However, we introduce some of the concepts and tooling support that is involved to help you plan and succeed with your IMS SOA project.

3.3.1 SOA-enabled Business Process Management

The IBM Business Process Management (BPM) Suite (as illustrated in Figure 3-4 on page 64) brings together capabilities from across IBM and includes a choice of two Foundational Offerings: the IBM WebSphere Dynamic Process Edition and the IBM FileNet® Active Content Edition. Optional Extended Value Offerings expand the value of the IBM BPM

Suite and include advanced analytics, BPM repositories, accelerators, adaptors, and collaborative tools.

WebSphere Dynamic Process Edition as an end-to-end foundational offering:

- ▶ Business user tools to visualize, understand, document, and simulate business processes that includes human work flows.
- ▶ A SOA-based process engine that is capable of dynamic execution of business processes based on business policies and service selection.
- ▶ A comprehensive Business Activity Monitor (BAM) that delivers a real-time view of business processes and operations.

IBM FileNet Active Content Edition consists of the following products:

- ▶ FileNet Business Process Manager increases process performance, reduces cycle times, and improves productivity by streamlining and improving complex processes.
- ▶ FileNet Business Activity Monitor (BAM) provides visibility into the effectiveness of business processes and performance.
- ▶ FileNet eForms enables easier electronic forms design, management, and processing as part of an enterprise content management platform.
- ▶ FileNet Business Process Framework (BPF) helps organizations to reduce business process management application development costs and time-to-market by accelerating development and deployment of business-specific solutions.

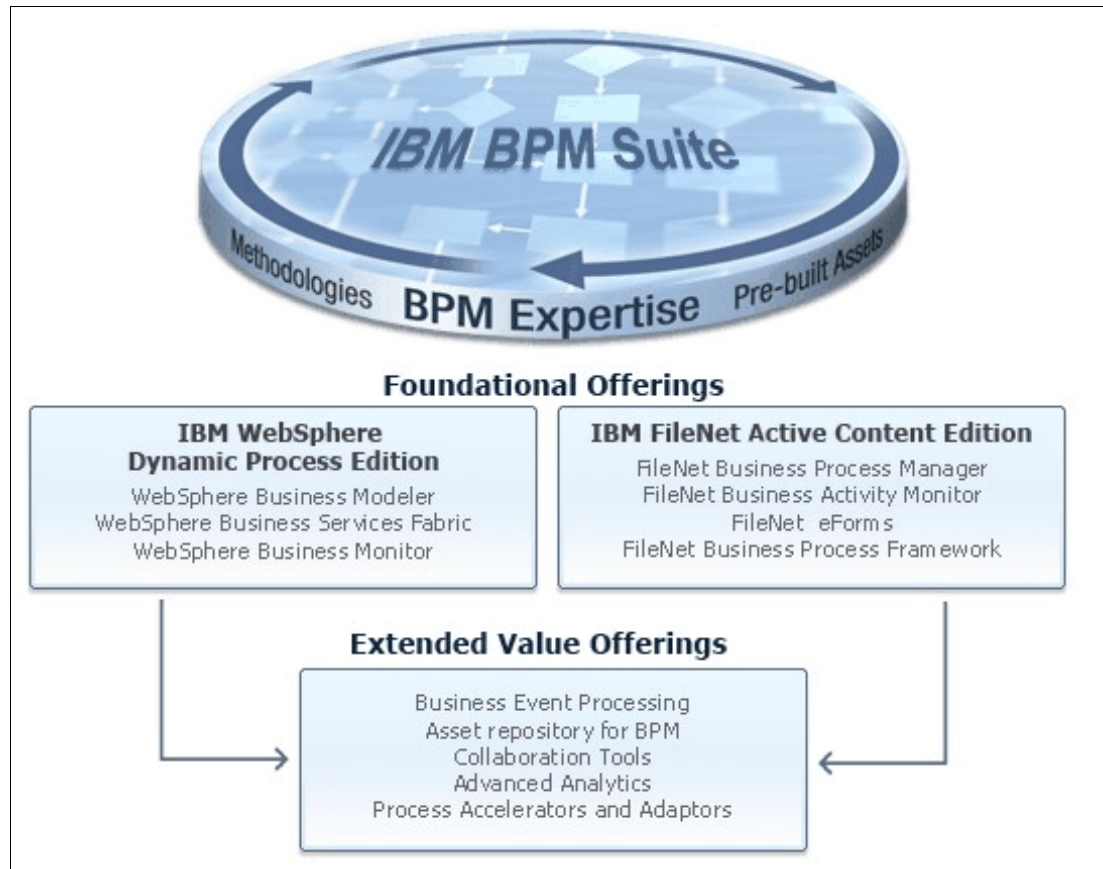


Figure 3-4 The IBM BPM Suite

For more information about BPM:

- ▶ BPM Enabled by SOA:
<http://www.ibm.com/software/info/bpm/>
- ▶ IBM WebSphere Dynamic Process Edition:
http://www.ibm.com/software/integration/wdpe/features/?S_CMP=rnav
- ▶ IBM FileNet Active Content Edition:
<http://www.ibm.com/software/data/content-management/products/process.html>

3.3.2 WebSphere Business Modeler

Documentation is the first step to understanding business process, which is usually followed by modifying the processes with the purpose of improving performance and efficiency. Business Processes are also needed to avoid anticipated bottle necks that were brought about during the process of business transformations.

WebSphere Business Modeler offers process modeling, simulation, and analysis capabilities to help business users understand, document, and deploy business processes.

To obtain more information about WBM:

- ▶ WebSphere Modeler User Group:
<http://www.ibm.com/developerworks/websphere/usergroups/>
- ▶ WebSphere Business Process Resources:
<http://www.ibm.com/developerworks/websphere/zones/bpm/>
- ▶ *Best Practices for Using WebSphere Business Modeler and Monitor*, REDP-4149-00
<http://www.redbooks.ibm.com/abstracts/redp4159.html?open>
- ▶ *IBM Business Process Management Reviewer's Guide*, REDP-44300
<http://www.redbooks.ibm.com/abstracts/redp4433.html?open>

3.3.3 WebSphere Service Registry and Repository

WebSphere Service Registry and Repository helps to manage service interactions and dependencies by handling policies, versioning, classification, and usage throughout the life of the service to promote optimal interaction of services in the SOA. It facilitates storing, accessing, and managing service information (service metadata) so you can easily select, invoke, govern, and reuse services.

For more information, access the Web site:

<http://www.ibm.com/software/integration/wsrr/>

3.3.4 WebSphere Integration Developer

WebSphere Integration Developer is the Eclipse tool to build the “importable” artifacts for the WebSphere Enterprise Service Bus Server (WESBS) and the WebSphere Process Server (WPS).

The processes to follow in WebSphere Integration Developer are comparable with Rational Application Developer V7. We develop within a module project and use the “external data” and “external service” options in menus.

The same types of connector objects are built as with Rational Application Developer V7, except that instead of databeans, the wizard builds *databinding* beans and a WSDL representation of the available service, so that with the “choreographer” it can be included in the mediation of the ESB or in the business flow in WPS.

The result of this wizard is a System Component (Import) as displayed in Figure 3-5.



Figure 3-5 IMS Import

If we examine the InteractionSpec properties of this artifact, we find the following elements:

- ▶ The interactionSpec class name
- ▶ The interaction verbs and the IMS request type
- ▶ The commit mode in use

Figure 3-6 displays the IMSInteractionSpec window.

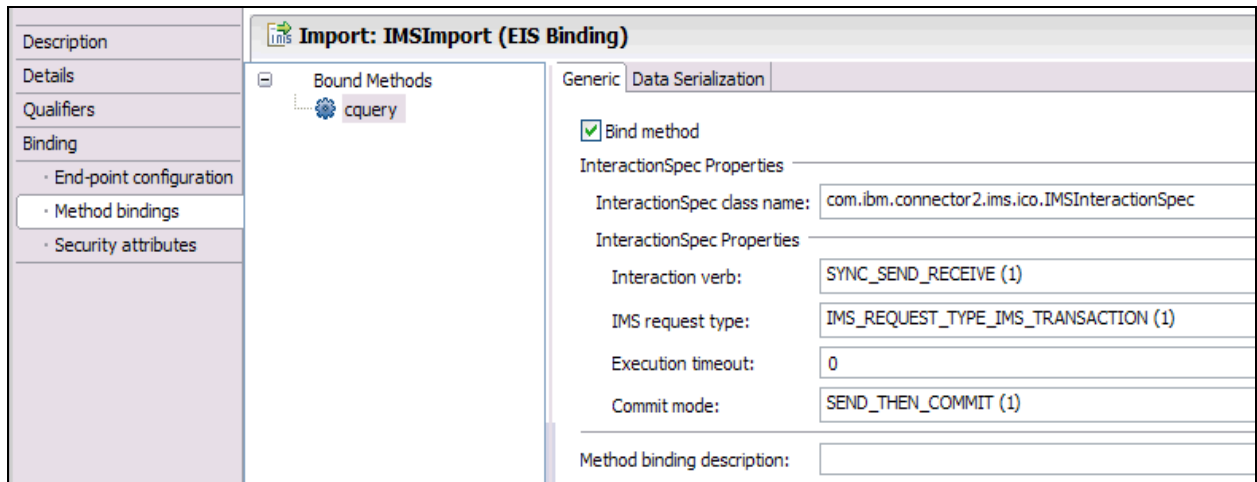


Figure 3-6 IMSInteractionSpec for method

The IMS transaction is called by invoking method “cquery” on the import and passing two databinding beans, one for input and output. Figure 3-7 on page 67 displays a graphical representation of the WSDL as it is shown by WebSphere Integration Developer.

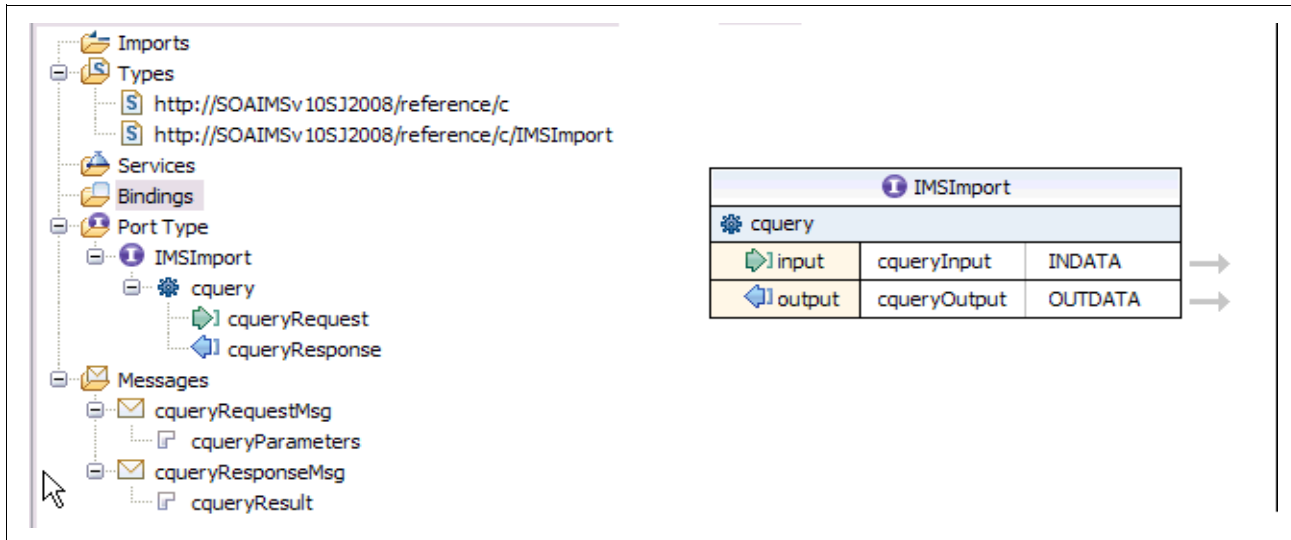


Figure 3-7 WSDL graphical view

WebSphere Integration Developer supports wizards that allow for easy and fast integration with IMS as an enterprise service, which includes the capability to import definitions from COBOL and mappings to IMS Message Format Service (MFS).

For more information, access the Web site:

<http://www.ibm.com/software/integration/wid/>

This concludes our discussion on IMS SOA tooling.

Creating, deploying, and managing IMS services

In part two of this publication, we describe many of the operational considerations that you must investigate during your IMS modernization projects.

In part two, we discuss:

- ▶ Chapter 4, “Example of deploying an IMS Service” on page 71
As mentioned in Chapter 1, “SOA and IMS: A powerful business combination” on page 1, we refer to earlier IBM Redbooks on IMS to the details that are associated with the deployment of IMS services, but will take you through an example to show how this sample deployment can solve a business situation.
- ▶ Chapter 5, “Securing IMS services” on page 85
There are additional component layers in SOA deployments that require securing. We introduce solutions that are available to secure your service flows.
- ▶ Chapter 6, “Problem management for IMS services” on page 105
As with security, the additional component layers that are associated with SOA require enhanced problem management.
- ▶ Chapter 7, “Performance considerations for IMS Services” on page 117
Regardless of how your applications are modernized, performance continues to be a key factor in Client satisfaction. There are a set of tools that you can use to monitor for important performance indicators, and we discuss those.



Example of deploying an IMS Service

To present an example of how the development and deployment process functions, we discuss a fictitious person's journey to deploying IMS SOAP Gateway on z/OS. For more information about SOAP Gateway, refer to Chapter 10, "The IMS SOAP Gateway" on page 203.

A corporation is undergoing a set of fundamental business changes in an effort to ultimately maximize profits. The corporation decided to adopt SOA principles to address the business and IT challenges that it faces. The corporate team is focusing on how to solve the challenges that are presented, by creating new customer accounts in a consistent manner throughout each of the sales channels. This SOA adoption initiative is known as the Account Open Project. An SOA approach allows for a more rapid implementation and greater flexibility for future changes that the business might need.

The case study that we describe in this chapter follows the journey of Samuel Johnson (Sam), an Application Developer, as he installs and deploys IMS SOAP Gateway on System z to be able to run the Installation Verification Procedure (IVP) as a proof-of-concept.

4.1 Background to SOAP Gateway

Service enabling IMS transactions using IMS SOAP Gateway Sam can enable selected IMS applications to become a Web service that can be consumed by the Account Open application or reused by other applications. With IMS SOAP Gateway, you can be flexible about how you want your data to be handled. XML data from the client can be shipped to IMS and stored directly in XML format using the IMS XML DB function, or it can be transformed into data bytes. Sam learns that the IMS SOAP Gateway consists of two main components:

- ▶ IMS SOAP Gateway deployment utility

Using the end-to-end deployment utility you can set up properties and create runtime code that IMS SOAP Gateway uses to enable IMS applications as Web services.

- ▶ IMS SOAP Gateway server

The server is used to process SOAP messages. It receives the SOAP message from the client application, converts it to an IMS input message, and sends it to IMS through IMS Connect. It then receives the output message from IMS, converts it to SOAP, and sends it back to the client.

4.1.1 SOAP Gateway development and deployment

Figure 4-1 depicts the environment used to assemble and deploy the service-enabled IMS transaction using the IMS SOAP Gateway. Rational Developer for System z is a major contributor to the development of the service.

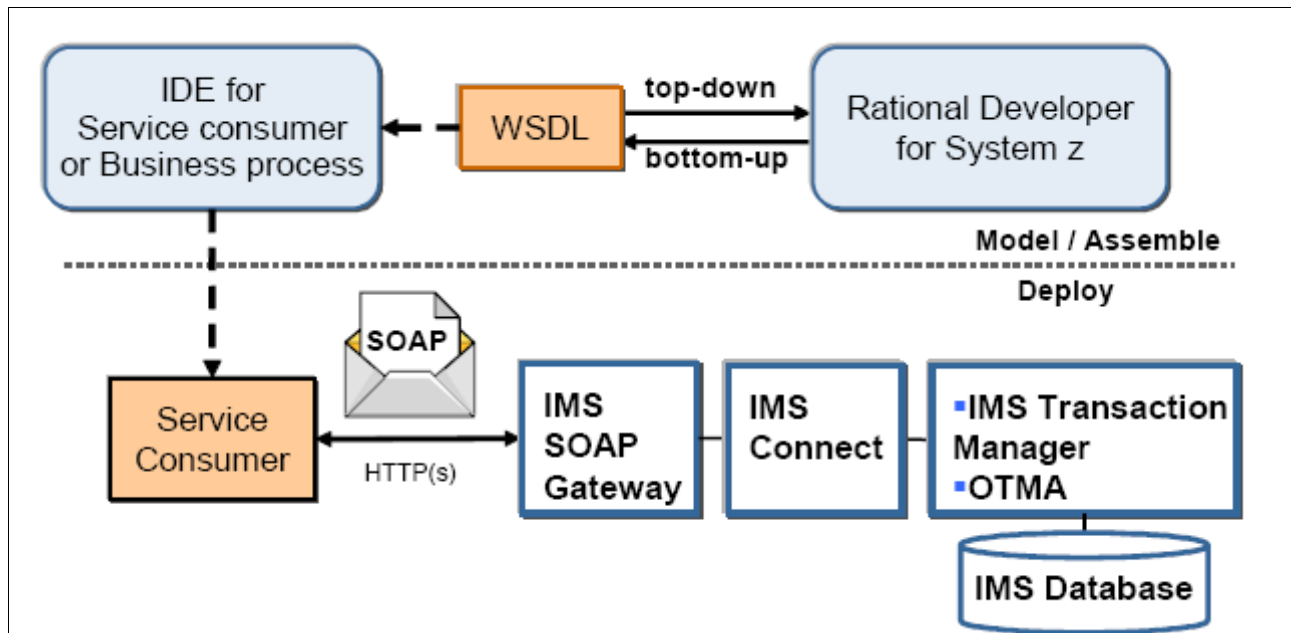


Figure 4-1 Service-enable IMS transactions using IMS SOAP Gateway

Sam starts by retrieving the financial records application artifacts, which includes COBOL programs and copybooks. The artifacts are imported into IBM Rational Developer for System z. Sam uses the tooling to generate the input and output XML converters, driver program, and service WSDL.

The generated WSDL is deployed to IMS SOAP Gateway and published to the IBM WebSphere Service Registry and Repository. This step is optional and requires that you purchase the IBM WebSphere Service Registry and Repository product.

In the next sections, we present the key considerations for service-enabled IMS transactions using an IMS SOAP Gateway.

4.2 Installing IMS SOAP Gateway on z/OS

Sam follows the sections below as a guide to install, set up, and verify IMS SOAP Gateway on z/OS. After IMS SOAP Gateway is properly up and running, there are sample applications (the general IMS Phonebook application for example) that can be deployed. Refer to section 4.4, “Accessing an IMS SOAP Gateway sample application” on page 84 for instructions about obtaining this sample application.

Other platforms, such as Windows, are supported, although z/OS is a preferred platform.

4.2.1 Installing and setting up

Installation prerequisites

There are a few simple prerequisites that must be observed:

- ▶ At least 225 cylinders of a 3390-3 data set are required to expand the downloaded PAX file.
- ▶ Superuser rights (the superuser, or root, is a special user account used for system administration) are needed when extracting the PAX file.

Important: The user ID that issues the PAX command must have UID(0) or READ access or higher to the following classes in the RACF FACILITY class:

- BPX.SUPERUSER
- BPX.FILEATTR.PROGCTL
- BPX.FILEATTR.APFBPX.FILEATTR.SHARELIB

To get started, the following files are required:

- ▶ The following MVS files:
 - (DFSHFSAL, DFSHFSMT, DFSMKDR, IMSSOAP, IMSSOAPZ)
- ▶ Installation program (imssoap101zos.pax).

Reference “Accessing the IMS SOAP Gateway Web site” on page 74 to obtain the files.

These files are later saved to a MVS data set:

- ▶ DFSMKDR: Creates the directories for inside the HFS.
- ▶ DFSHFSAL: Allocates the HFS.
- ▶ DFSHFSMT: Mounts the HFS.
- ▶ IMSSOAP: Creates the PROC to start IMS SOAP Gateway.
- ▶ IMSSOAPZ: This the configuration member for the IMSSOAP PROC.

One pax file is later saved to a temporary HFS directory:

- ▶ imssoap101zos.pax: Installation program

Accessing the IMS SOAP Gateway Web site

The Web site for the IMS SOAP Gateway site is:

<http://www.ibm.com/software/data/ims/soap/>

To access the IMS Soap Gateway Web site:

1. Type the following address:
<http://www.ibm.com/software/data/ims/soap/>
2. On this page, there is a link to register for the 'IMS SOAP Gateway download Web site. Select it.
3. This is the SOAP Gateway sign on window. Select **Sign in** and use Username: imssouser@us.ibm.com and password: imssouser
4. On the next window, select **IMS SOAP Gateway V10.1 with Rational Developer for System Z V7.1.1**.
5. Click **Continue**.
6. Select **I agree**.
7. Click **I confirm**.
8. After you are at the download page, select the six files mentioned under the z/OS section. Figure 4-2 presents this list.

z/OS	
<input checked="" type="checkbox"/>	MVS Template File DFSHFSAL (3KB)
<input checked="" type="checkbox"/>	MVS Template File DFSHFSMT (4KB)
<input checked="" type="checkbox"/>	MVS Template File DFSMKDR (1KB)
<input checked="" type="checkbox"/>	MVS Template File IMSSOAP (3KB)
<input checked="" type="checkbox"/>	MVS Template File IMSSOAPZ (3KB)
<input checked="" type="checkbox"/>	IMS SOAP Gateway V10.1 Installation Program for zOS imsssoap101zos.pax (115MB)

Figure 4-2 Files needed from the IMS SOAP Gateway Web site

Note: The IMS SOAP Gateway download site uses the Download Directory to download the files. After the download is complete, the files must be located on your workstation under the directory specified.

4.2.2 Preparing for installation

At this point, all of the files must be downloaded onto your workstation. The next steps guide you through copying the MVS files from your workstation to a MVS-partitioned data set using FTP. To modify the JCL we use OMVS (UNIX System Services).

Note: A FTP client or the Windows DOS command can be used to get your files onto the OMVS Admin. For this example, we use the Windows DOS command.

The FTP instructions for the MVS files are:

1. Initialize the Windows DOS command.
2. Change the directory to the download directory location:
 - Example: cd C:\DownloadDirectory
3. To verify that the files are located in the directory, issue dir.
4. Type ftp.
5. Enter OMVSAdmin username: <username>.
6. Enter OMVSAdmin password: <pwd>.
7. Change to the destination directory location:
 - Example: cd 'user.private.proclib'
8. Type put DFSHFSAL:
 - After the file transfers, the message Transfer completed successfully is displayed.
9. Repeat step 8 to transfer the remaining MVS files (DFSHFSMT, DFSMKDR, IMSSOAP, and IMSSOAPZ). See Figure 4-3 for a sample of the output of the command stream.

```
C:\DownloadDirector\SOAPPparts>dir
Volume in drive C has no label.
Volume Serial Number is 8017-F8A8

Directory of C:\DownloadDirector\SOAPPparts

08/18/2008 02:06 PM <DIR>          .
08/18/2008 02:06 PM <DIR>          ..
08/18/2008 01:31 PM                2,319 DFSHFSAL
08/18/2008 01:31 PM                3,302 DFSHFSMT
08/18/2008 01:31 PM                   537 DFSMKDR
08/18/2008 01:32 PM                4,237 dlmgr.pro
08/18/2008 01:31 PM                2,586 IMSSOAP
08/18/2008 01:32 PM   115,799,040 imsssoap101zos.pax
08/18/2008 01:31 PM   116,017 IMSSOAPGwU101iFix1.zip
08/18/2008 01:31 PM                3,027 IMSSOAPZ
                8 File(s)  115,931,065 bytes
                2 Dir(s)  61,389,250,560 bytes free

C:\DownloadDirector\SOAPPparts>ftp
ftp> open ec01116.svl.ibm.com
Connected to ec01116.svl.ibm.com.
220-FTPDI IBM FTP CS UIR7 at EC01116.svl.ibm.com, 21:53:24 on 2008-08-18.
220 Connection will close if idle for more than 5 minutes.
User (ec01116.svl.ibm.com:(none)): omvsadm
331 Send password please.
Password:
230 OMUSADM is logged on. Working directory is "OMUSADM.".
ftp> cd 'user.private.proclib'
250 The working directory "USER.PRIVATE.PROCLIB" is a partitioned data set
ftp> put DFSHFSAL
200 Port request OK.
125 Storing data set USER.PRIVATE.PROCLIB(DFSHFSAL)
250 Transfer completed successfully.
ftp: 2319 bytes sent in 0.03Seconds 74.81Kbytes/sec.
ftp> put DFSHFSMT
200 Port request OK.
125 Storing data set USER.PRIVATE.PROCLIB(DFSHFSMT)
250 Transfer completed successfully.
ftp: 3302 bytes sent in 0.01Seconds 220.13Kbytes/sec.
ftp> put DFSMKDR
200 Port request OK.
125 Storing data set USER.PRIVATE.PROCLIB(DFSMKDR)
250 Transfer completed successfully.
ftp: 537 bytes sent in 0.00Seconds 537000.00Kbytes/sec.
ftp>
```

Figure 4-3 Windows DOS FTP to MVS files

The FTP instructions for the .pax file are:

1. Initialize the Windows DOS Command.
2. Change the directory to the download directory location:
 - Example: `cd C:\DownloadDirectory`
3. To verify that the files are located in the directory, issue `dir`.
4. Type `ftp`.
5. Enter OMVSAdmin username: `<username>`.
6. Enter OMVSAdmin password: `<pwd>`.
7. Change to the destination directory location:
 - Example: `cd '/u/oeusr01'`
8. Type `binary`.
 - **Important:** The file transfers in binary mode from ASCII to OMVS.
9. Type `put imsssoap101zos.pax`:
 - After the file transfers, the message `Transfer completed successfully` is displayed. See Figure 4-4 for a sample of the output of the command stream.

```
C:\DownloadDirector\SOAparts>ftp
ftp> open ec01116.svl.ibm.com
Connected to ec01116.svl.ibm.com.
220-FTPD1 IBM FTP CS U1R7 at EC01116.svl.ibm.com, 22:42:55 on 2008-08-18.
220 Connection will close if idle for more than 5 minutes.
User (ec01116.svl.ibm.com:(none)): omvsadm
331 Send password please.
Password:
230 OMVSADM is logged on. Working directory is "OMVSADM.".
ftp> cd /u/oeusr01
250 HFS directory /u/oeusr01 is the current working directory.
ftp> binary
200 Representation type is Image
ftp> put imsssoap101zos.pax
200 Port request OK.
125 Storing data set /u/oeusr01/imsssoap101zos.pax
250 Transfer completed successfully.
ftp: 115799040 bytes sent in 10.38Seconds 11161.35Kbytes/sec.
ftp>
```

Figure 4-4 Windows DOS FTP - PAX File

4.2.3 Setting up

At this point, all of the files are downloaded and transferred to the MVS environment. The next step is to set up the environment by modifying the JCL.

Each JCL template file contains a comment section that describes the changes that are needed. Make the necessary changes, based on your environment, in the following files:

- ▶ DFSMKDR
- ▶ DFSHFSAL
- ▶ DFSHFSMT

Editing DFSMKDR

The DFSMKDR job creates the mount point and the directories under the mount point. The mount point is dependent on what the `-PathPrefix` parameter is set to:

- ▶ If `-PathPrefix` is set to `/u`, the mount point is `/u/oeusr01/usr/ims/imsssoap101zos`
- ▶ If `-PathPrefix` is set to `null`, the mount point is `/oeusr01/ims/imsssoap101zos`

Tip: If you are copying and pasting text from your workstation, you are copying from ASCII format and pasting into EBCDIC format on MVS, so single and double quotation marks sometimes are deleted.

Editing DFSHFSAL

Edit the following information prior to submitting DFSHFSAL:

1. Change the job card to meet your systems requirements. See Example 4-1 for a sample job card for DFSHFSAL.

Example 4-1 Sample job card for DFSHFSAL

```
//DFSHFSAL JOB ,  
// CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),  
// TIME=(9),USER=USRT001,PASSWORD=ALL1SDUN,  
// REGION=32M
```

2. Change *hfsdsn* to the new file system.
3. Change the *valid* to the volser id of the disk that contains the IMS data sets. See Example 4-2 for a sample of the valid modification.

Example 4-2 Sample of HFSDSN-VOLID

```
ALLOCATE  
  DSNAME('IMSTESTL.EC01116.HFS') - hfsdsn  
  RECFM(U) -  
  LRECL(0) -  
  BLKSIZE(32760) -  
  DSORG(PO) -  
  VOLUME(IMSXB1) - valid  
  DSNTYPE(HFS) -  
  NEW CATALOG -  
  SPACE(600,30) CYL -  
  DIR(200) -  
  UNIT(SYSALLDA)
```

4. Verify the JCL job submission, as shown in Example 4-3.

Example 4-3 Successful submission message

```
15.38.17 JOB000072 $HASP373 DFSHFSAL STARTED - INIT 8 - CLASS A - SYSSTL1  
15.38.17 JOB000072 SMF000i DFSHFSAL ALLOCATE IDCAMS 0000  
15.38.17 JOB000072 $HASP395 DFSHFSAL ENDED
```

Editing DFSHFSMT

Edit the following information prior to submitting DFSHFSMT:

1. Review the control statements before submitting the job.
2. Change the job card to meet your system requirements.
3. Change *hfsdsn* to the new file system.
4. Change the *-MountPointPath-* to the appropriate high-level directory name.
5. Submit the JCL.
6. Verify the submission was successful.

Example 4-4 shows a listing of the mount point job.

Example 4-4 Sample mount point

```
//MOUNT1 EXEC PGM=IKJEFT01
//*-----*/
/* This step defines the mount point directory and mounts */
/* the HFS. */
/*-----*/
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
MOUNT FILESYSTEM('IMSTESTL.EC01116.HFS') +
MOUNTPOINT('/u/oeusr01') TYPE(HFS) MODE(RDWR)
//
```

Note: You must specify the mount point. In this example we use /u/oeusr01

4.2.4 Installing the SOAP Gateway

We now map the SOAP Gateway installation process.

Extracting the PAX file

From OMVS, change the directory to the directory that was created by the DFSMKDR job. Issue the following command to extract the PAX file:

```
pax -pxa -ruvf /u/oeusr01/imssoap101zos.pax
```

In this command, /u/oeusr01/ is the directory into which the PAX file was uploaded.

Setting up RACF for the JCL procedures

To secure the environment, RACF is used. Table 4-1 presents RACF variable selections.

Table 4-1 RACF variables set-up

Substitution variable	Description	Example
jjjjj	Name of the started PROC	IMSSOAP
hhhhh	Home directory for the OMVS segment. The home directory must exist. This directory is the default location for the IMS SOAP Gateway trace file, imsssoap.log.	/u/oeusr01
nnnnn	OMVS user ID. If UID(0) is used, no permission is required on the data sets. Otherwise, the mount point directory must have an access privilege of 755.	12345
ggggg	Default group for the specified user ID.	SYS1

To set up RACF for the JCL procedures:

1. In the ISPF panel, select option 6.
2. Define the user ID and OMVS segment in the home directory if they do not exist yet. The home directory must already exist.

If you are defining a new user ID or making changes to an existing user, such as adding an OMVS segment (in PROC), issue the following command:

```
AU IMSSOAP DFLTGRP(SYS1) OMVS(UID(0) HOME('/u/oeusr01') PROGRAM('/bin/sh'))
```

3. Define the started task:

```
RDEF STARTED IMSSOAP.* STDATA(USER(IMSSOAP) GROUP(SYS1))
```

4. Refresh the started class:

```
SETR RACLIST(STARTED) REFR
```

See Example 4-5 for a screen shot of the ISPF TSO command panel after the security related commands were issued.

Example 4-5 ISPF menu after commands and are submitted

```
Menu List Mode Functions Utilities Help
-----
                                USRT001
                                ISPF Command Shell
Enter TSO or Workstation commands below:

====>
```

Place cursor on choice and press enter to Retrieve command

```
=> RDEF STARTED IMSSOAP.* STDATA(USER(IMSSOAP) GROUP(SYS1))
=> SETR RACLIST(STARTED) REFR
=> AU IMSSOAP DFLTGRP(SYS1) OMVS(UID(0) HOME('/u/oeusr01') PROGRAM('/bin/sh'))
```

Copying the JZOS load module to a PDSE

You must copy the JZOS load module to a partitioned data set extended (PDSE) address space. A PDSE is a data set that is defined with a data set type of library and that can be accessed from both MVS and OMVS:

1. Allocate the PDSE data set.
2. From an OMVS command prompt, change the directory to the mount point that was created by the DFSMKDR job.

Append the mount point with the imsssoap101/Java/mvstools:

```
cd /u/oeusr01/imsssoap101/Java/mvstools
```

3. Copy the module to a load library that is defined as a PDSE, for example, to copy the JVMLDM14 load module to IMSTESTL.JZOS.LOADLIB, issue the following command:

```
cp -X JVMLDM14 '//IMSTESTL.JZOS.LOADLIB(JVMLDM14)'
```

Modifying the JCL procedures for starting IMS SOAP Gateway

Each JCL template file contains a comment section that describes the changes that are needed. Make the necessary changes, based on your environment, in the following files:

- ▶ IMSSOAP
- ▶ IMSSOAPZ

To modify the JCL procedures for starting IMS SOAP Gateway:

1. You must modify the USER.PRIVATE.PROCLIB(IMSSOAP) JCL procedures (PROC) for starting IMS SOAP Gateway and the configuration member that is used by the USER.PRIVATE.PROCLIB(IMSSOAPZ) proclib library. See Example 4-6 for the IMSSOAP procedure.

Example 4-6 Modified JCL for the IMSSOAP procedure

```
//IMSSOAP PROC REGSIZE='0M',
//  JavaCLS='org.apache.catalina.startup.Bootstrap',
//  ARGS='start',
//  LIBRARY='IMSTESTL.JZOS.LOADLIB',      < PDSE FOR JZOSVM MODULE
//  VERSION='14',                        < VERSION OF JZOSVM MODULE
//  LOGLVL='',                            < DEBUG LVL: +I(INFO) +T(TRC)
//  PARM='',
//  LEPARM=''
//IEFPROC EXEC PGM=JVMLDM&VERSION,REGION=&REGSIZE,
//  PARM='&LEPARM/&LOGLVL &JavaCLS &ARGS &PARMS'
//STEPLIB DD DSN=&LIBRARY,DISP=SHR
//SYSPRINT DD SYSOUT=*                  < SYSTEM STDOUT
//SYSOUT DD SYSOUT=*                   < SYSTEM STDERR
//STDOUT DD SYSOUT=*                   < Java SYSTEM.OUT
//STDERR DD SYSOUT=*                   < Java SYSTEM.ERR
//CEEDUMP DD SYSOUT=*
//ABNLIGNR DD DUMMY
//*
//*
//STDENV DD DISP=SHR,DSN=USER.PRIVATE.PROCLIB(IMSSOAPZ)
```

2. Modify the configuration member that is used by the SYS1.PROCLIB(IMSSOAPZ) PROC library. Use the following substitution variable table to modify the SYS1.PROCLIB(IMSSOAPZ) ROC library. See Example 4-7 for the substitution variable table.

Example 4-7 Substitution variable table

```
# script to set up for imsssoap gateway
export IMSSOAP_DIR=/u/oeusr01
export CATALINA_HOME="${IMSSOAP_DIR}"/server
export IMSSOAP_HOME="${IMSSOAP_DIR}"/server/Webapps/imsssoap
export Java_HOME="${IMSSOAP_DIR}"/Java
export PATH=/bin:"${Java_HOME}"/bin:

LIBPATH=/lib:/usr/lib:"${Java_HOME}"/bin
LIBPATH="${LIBPATH}:"${Java_HOME}"/bin/classic
export LIBPATH="${LIBPATH}:"

CLASSPATH="${Java_HOME}/lib/tools.jar"
CLASSPATH="${CLASSPATH}:"${CATALINA_HOME}/bin/bootstrap.jar"
export CLASSPATH="${CLASSPATH}:"

# Set JZOS specific options
# Use this variable to specify encoding for DD STDOUT and STDERR
#export JZOS_OUTPUT_ENCODING=Cp1047
# Use this variable to prevent JZOS from handling MVS operator commands
#export JZOS_ENABLE_MVS_COMMANDS=false
```

```

# Use this variable to supply additional arguments to main
#export JZOS_MAIN_ARGS=""

# Configure JVM options
# Note that Tomcat requires default ASCII file.encoding
IJO="-Xms64m -Xmx128m"
# IJO="$IJO -verbose:gc"
IJO="$IJO -Dfile.encoding=ISO8859-1"
IJO="$IJO -Dcatalina.base=${CATALINA_HOME}"
IJO="$IJO -Dcatalina.home=${CATALINA_HOME}"
IJO="$IJO -Djava.io.tmpdir=${CATALINA_HOME}/temp"
IJO="$IJO -Djava.endorsed.dirs="
IJO="${IJO}${CATALINA_HOME}/common/endorsed"
export IBM_Java_OPTIONS="$IJO "

export Java_DUMP_HEAP=false
export Java_PROPAGATE=NO
export IBM_Java_ZOS_TDUMP=NO

```

You successfully deployed IMS SOAP Gateway on z/OS. The next step is to start the IMS SOAP Gateway Server from the OMVSAdmin Console.

Type: `iogdeploy.sh`, and the Deployment utility menu is displayed, as shown in Example 4-8.

Example 4-8 IMS SOAP Gateway Deployment utility menu on z/OS

```

=====
Enable your IMS application as a Web service :
  Task 1: Enable your IMS application as a Web service provider

Administrative tasks ( Type "cancel" to return to the main menu):
  Task 2: Start IMS SOAP Gateway (not applicable for z/os)
  Task 3: Stop IMS SOAP Gateway (not applicable for z/os)
  Task 4: Update IMS SOAP Gateway properties
  Task 5: Create, Update or View correlator properties for Web Service
  Task 6: Create, Update, Delete or View connection bundle
  Task 7: Deploy the WSDL file
  Task 8: Generate Java client code
  Task 9: Undeploy Web service
  Task 10: Enable your IMS application to access an external Web Service
  Task 11: Exit deployment utility
=====
> Enter your selection here:

```

4.2.5 Post installation steps

The post installation steps are:

- ▶ Apply any latest iFix that is posted on the IMS SOAP Gateway download Web page. Follow the installation instructions that are provided with the iFix.

- ▶ To verify the successful installation of IMS SOAP Gateway, follow the steps described in “Setting up IMS Connect and IMS OTMA” on page 83. Then you can verify the installation by using the IMS SOAP Gateway Installation Verification Program (IVP):

- a. Open an Internet browser (Internet Explorer, or Mozilla Firefox).
- b. Enter the URL: `http://<hostname>:<portnumber>/imssoap/` where ‘portnumber’ is specified by you.

The IMS SOAP Gateway Installation Verification Program(IVP) page loads.

- c. Click **Submit**.

Figure 4-5 shows you what you can expect to see.

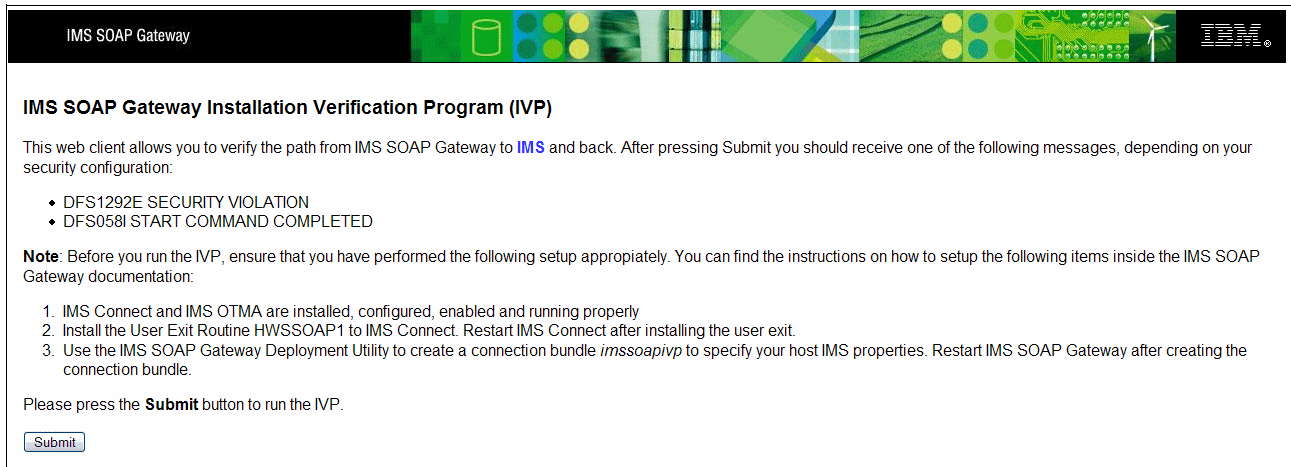


Figure 4-5 The IMS Soap Gateway IVP page

- d. After you click **Submit**, a message is displayed, as shown in Figure 4-6.

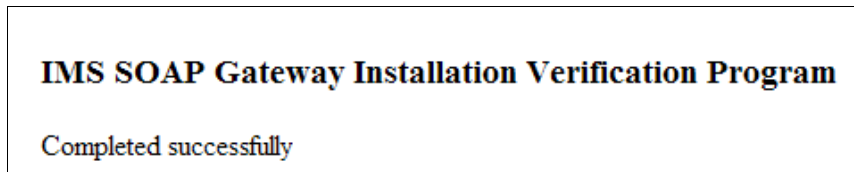


Figure 4-6 IVP success message

- ▶ If you are upgrading from IMS SOAP Gateway Version 9.x, because of the correlator properties changes in IMS SOAP Gateway Version 10, you must manually modify your correlator files.

Now that Sam installed IMS SOAP Gateway and completed his proof-of-concept study, he can now begin the SOA adoption initiative known as the Account Open Project.

To be able to practice how to enable an IMS application as a Web service running in an IMS SOAP Gateway on Microsoft Windows, go to:

http://www.ibm.com/developerworks/downloads/emsandbox/systemz/architecture/?S_TACT=105AGX15&S_CMP=EMDOC

This is the home page for the IBM enterprise modernization sandbox for System z architecture. Select the link for the IMS SOAP Gateway exercise.

4.3 Setting up IMS Connect and IMS OTMA

IMS SOAP Gateway uses IMS Connect and IMS OTMA to invoke an IMS transaction as a Web service. For IMS SOAP Gateway to access IMS transactions, you must configure IMS OTMA and IMS Connect.

IMS Connect and IMS OTMA must have the following tasks completed in order for IMS SOAP Gateway to access IMS transactions:

- ▶ IMS Connect and IMS OTMA are installed, configured, enabled, and running properly. For IMS Connect:
 - a. Configure IMS Connect with User Exit Routine HWSSOAP1. User Exit Routine HWSSOAP1 is available for IMS Connect Version 9, through APARs PK24912 and PK29938, which are included in PTF UK17000.
 - b. Select the data transformation process. You can handle the conversion of XML messages between IMS SOAP Gateway and IMS with one of the following:
 - i. IMS Connect can be configured to convert the XML messages that are received from IMS SOAP Gateway into an IMS transaction format. If you use this method for data conversion, the IMS Connect XML Adapter must be configured. Instructions about how to configure the IMS Connect XML Adapter are located in the ++ HOLD card data in the IMS Connect APAR for the Adapter Task Manager (PTF UK17000, including APAR PK24912 and APAR PK29938) and in the *IMS Version 10 Communications and Connections Guide*, SC18-9703.
 - ii. XML messages can be sent directly to IMS, and the IMS application must be modified to accept the XML input messages and to return a XML output message.
- ▶ IMS Connect and OTMA are set up with the correct level of security, for example, if you have security enabled for IMS Connect and IMS OTMA, provide the valid security properties (such as user ID, password and group name) to IMS SOAP Gateway. This task is performed later using the IMS SOAP Gateway Installation Verification Program (IVP).
- ▶ The IMS transaction and the program that is associated with the IMS transaction were started and are running properly.
- ▶ Optional: Set up IMS Connect SSL ports:
 - a. Refer to the IMS Connect SSL connections topic in the IMS Version 10 library on how to set up the SSL configuration in IMS Connect.
 - b. On your z/OS system, verify that the SSL ports are active using the z/OS Modify Command, Control Center, or MVS reply-to number, for example, if you use MVS reply-to number, use the VIEWHWS command. This command displays a list of the SSL ports and details their status. Ensure that some of the ports have the suffix "s" and are in the ACTIVE state.

Note: For SSL support, set GSK_PROTOCOL_TLSV1 to ON. For more information, refer to *IMS Version 10 Communications and Connections Guide*, SC18-9703.

Post-installation tasks:

- ▶ Apply any latest iFix that is posted on the IMS SOAP Gateway download Web page. Follow the installation instructions that are provided with the iFix.
- ▶ You are ready to verify the installation by using the IMS SOAP Gateway Installation Verification Program (IVP).

4.4 Accessing an IMS SOAP Gateway sample application

In this section, we guide you through the steps of accessing the phonebook sample for IMS SOAP Gateway:

1. From your Internet browser enter: **http://www.ibm.com/software/data/ims/soap/**
2. Locate and select the **samples** reference.
3. Use Username: **imssoapuser@us.ibm.com** and password: **imssoapuser**.
4. Select **IMS SOAP Gateway V10.1 with Rational Developer for System Z V7.1.1**.
5. Click **Continue**.
6. Select **I agree**.
7. Click **I confirm**.

After you are at the download page, you will see the six z/OS files that we mentioned in 4.2.1, “Installing and setting up” on page 73. The iFIX selection, as mentioned in 4.2.5, “Post installation steps” on page 81, follows them. Then, as shown in Figure 4-7, the available selections of samples are available for download.

Samples	
<input type="checkbox"/>	IMS SOAP Gateway Phone Book Sample IMSSGPB101Sample.zip (1.45MB)
<input type="checkbox"/>	IMS SOAP Gateway sample steps for asynchronous callout setup IMSSGPB101AsyncCalloutSample.zip (1.45MB)

Figure 4-7 IMS SOAP Gateway phone book sample

Note: IMS SOAP Gateway download site uses the Download Directory to download the files. After the download is complete, the files should be located on your workstation under the directory specified.

This completes our discussion regarding an example of deploying an IMS service.



Securing IMS services

Companies that adopt SOA require a security model that enables secure business transactions across enterprises and the Internet. The security domain includes topics, such as authentication, single sign-on, authorization, federated identity management, and user provisioning. We discuss these topics in this chapter.

5.1 Security ISO and standards

IT security objectives are specified in ISO Standard 7498-2 (ISO reference model for open systems interconnection (OSI)):

- ▶ Identification: The ability to assign an identity to the entity that accesses the system, which can be items, such as an user ID, UID, or principal in the Java EE security model.
- ▶ Authentication: The process of validating the identity that the accessing entity claims. Authentication information is generally called credentials and can include:
 - Accessor's name and password
 - Token provided by a trusted party, such as a *Kerberos ticket* (Kerberos is a computer network authentication protocol).
 - A x.509 certificate, which is a standard for a *public key infrastructure* (PKI)
 - Lightweight Third-Party Authentication (LTPA) token
- ▶ Authorization: The process of checking whether an asserted (already authenticated) identity has access to a requested resource.
- ▶ Integrity: Integrity ensures that transmitted or stored information was not altered in an unauthorized or accidental manner.
- ▶ Confidentiality: Refers to the concept that an unauthorized party cannot obtain the meaning of the transferred or stored data.
- ▶ Auditing: With auditing, you capture and record security-related events, so that they can be exposed and analyzed after the fact.

5.2 Layers of SOA security

The layers of SOA security are:

- ▶ On the System z platform: *System Authorization Facility* (SAF) is an umbrella term for IBM RACF, CA Top Secret, or CA ACF2 security software and databases that can be present in a z/OS operating environment. These security products contain information about users, groups of users, resources, and user's or group's access to those resources. The purpose of these products is to provide authentication and access control for the OS/390® and z/OS environment.
- ▶ JVM 5: The Java Virtual Machine (JVM) security model provides a layer of security above the operating system layer, for example, memory thread protection from unauthorized access.
- ▶ CORBA security: Any calls made through *Object Request Broker* (ORB) middleware software are invoked over a *Secure Association Service* (SAS) or *Common Secure Interoperability Protocol Version 2* (CSIv2) layer that sets up the security context and the necessary quality of protection. After the session is established, the call is passed up to the enterprise bean layer.
- ▶ Java EE security: The security collaborator enforces Java EE-based security policies and supports Java EE security APIs.
- ▶ WebSphere security: WebSphere security enforces security policies and services in a unified manner on access to Web resources and enterprise beans. It consists of WebSphere security technologies and features to support the needs of a secure enterprise environment.

Note: *Security Attribute Service* (SAS) in CSIv2 is different from SAS (Security Association Service), which is an IBM proprietary protocol used in WebSphere Version 6. Also WebSphere V6 SAS on the distributed platform is different than z/SAS for WebSphere z/OS.

Figure 5-1 shows you the security layers that we previously mentioned.

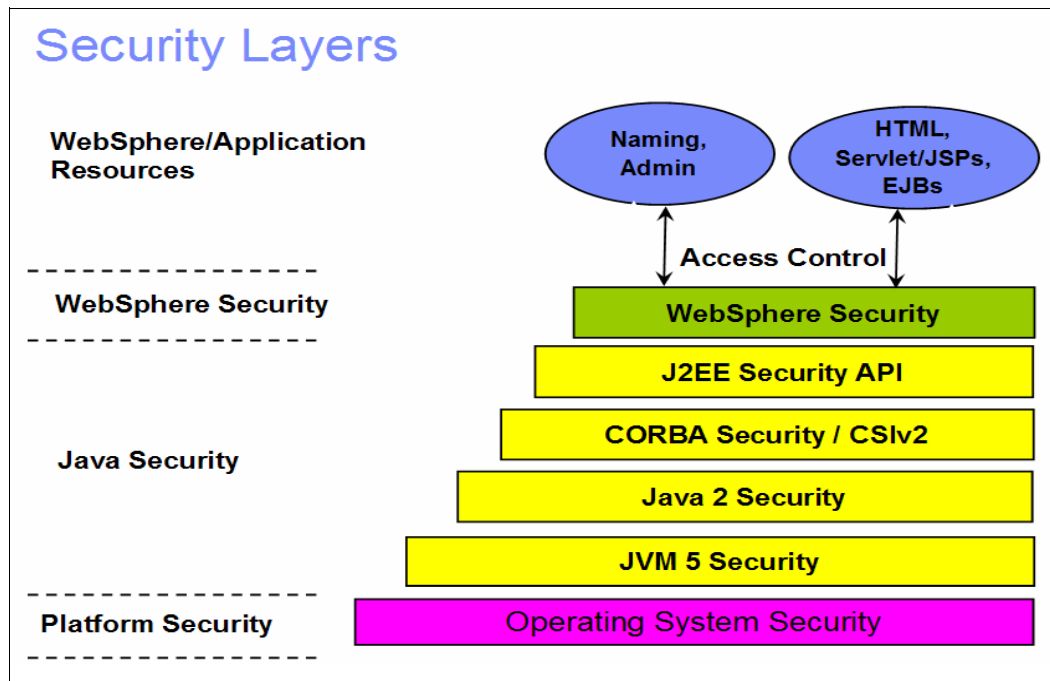


Figure 5-1 Security layers

5.3 Security management

Security management's goal is to provide an end-to-end security model to allow Java EE applications to access an EIS securely. In EIS applications, components request a connection to an EIS resource. As part of this connection, the EIS can require a sign-on for the requester to access the resource.

The security information is supplied through the following components:

- ▶ Application component (component-managed EIS sign-on)
- ▶ Application server (container-managed EIS sign-on)
- ▶ Connection factory custom properties

As we discuss these components, we use the *IMS TM Resource Adapter* (TMRA) security support to elaborate. The IMS TMRA supports the Java EE security model.

5.3.1 Component-managed EIS sign-on

In the component-managed sign-on approach, the application component code manages EIS sign-on by including code that performs the sign-on process to an EIS.

In component-managed sign-on, an application component is responsible for passing the needed sign-on security information to the resource to the getConnection() method. For example, security information might be a user name and password, as shown in Example 5-1.

Example 5-1 Component managed sign-on

```
// Obtain a connection
properties.setUserName("...");
properties.setPassword("...");
Javax.resource.cci.Connection cx =
    cxf.getConnection(properties);
```

Figure 5-2 displays the high-level flow of sign-on security information. The IMS TM Resource Adapter passes the security information to IMS Connect as required for authentication. IMS Connect in turn passes the security information to IMS OTMA for use in authorization.

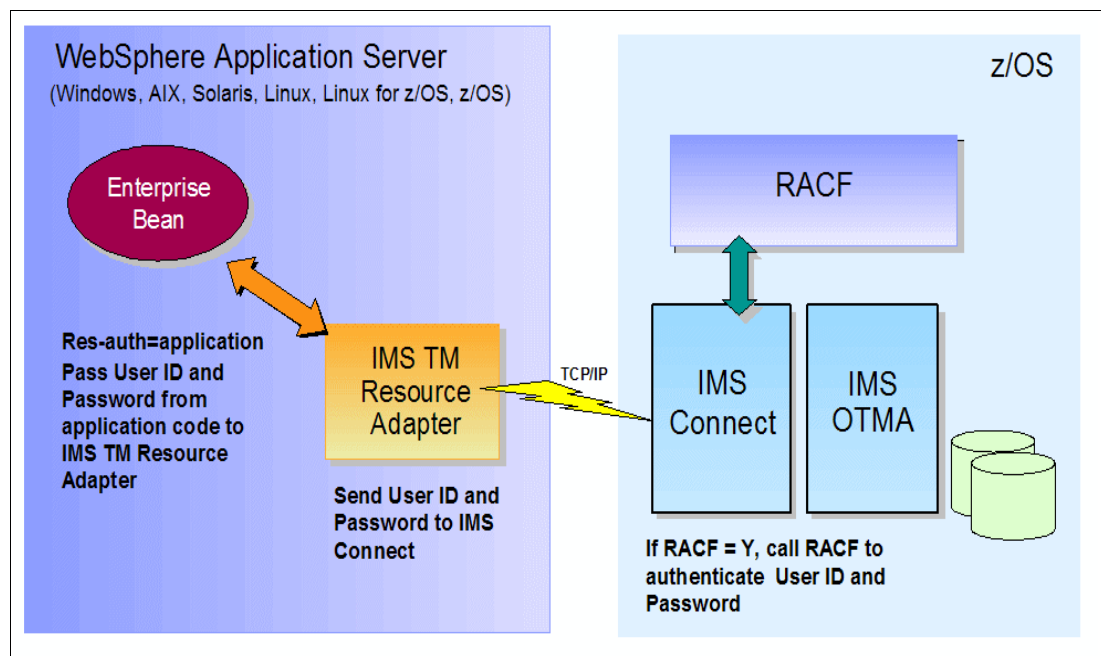


Figure 5-2 Component-managed EIS sign-on

5.3.2 Container-managed EIS sign-on

In the container-managed sign-on approach, the application component allows the container to take the responsibility of configuring and managing the EIS sign-on. Here is a sample of the code that is issued in the application's deployment descriptor:

```
Set <res-auth>Container</res-auth>
```

Applications developed without consideration for user authentication simplifies the environment because:

- ▶ There is no need to recompile code when changing security information
- ▶ Security information is centralized:
 - Easier to maintain security information
 - Security information is hidden from users

5.3.3 Connection factory custom properties

Java Message Service (JMS) provides a method of separating the application from the data transport layer. A connection represents a communication link between the application and the messaging server. The Java classes first use a connection factory to connect to the queue or topic, and then use populate and send to publish the messages. On the receiving side, the clients then receive or subscribe to the messages. Example 5-2 provides an example of a getConnection:

Example 5-2 ConnectionFactory getConnection example

```
Connection connection = ConnectionFactory.getConnection();
```

5.4 The WebSphere security mechanism

Figure 5-3 shows all of the security mechanisms that are available on the WebSphere Application Server for z/OS.

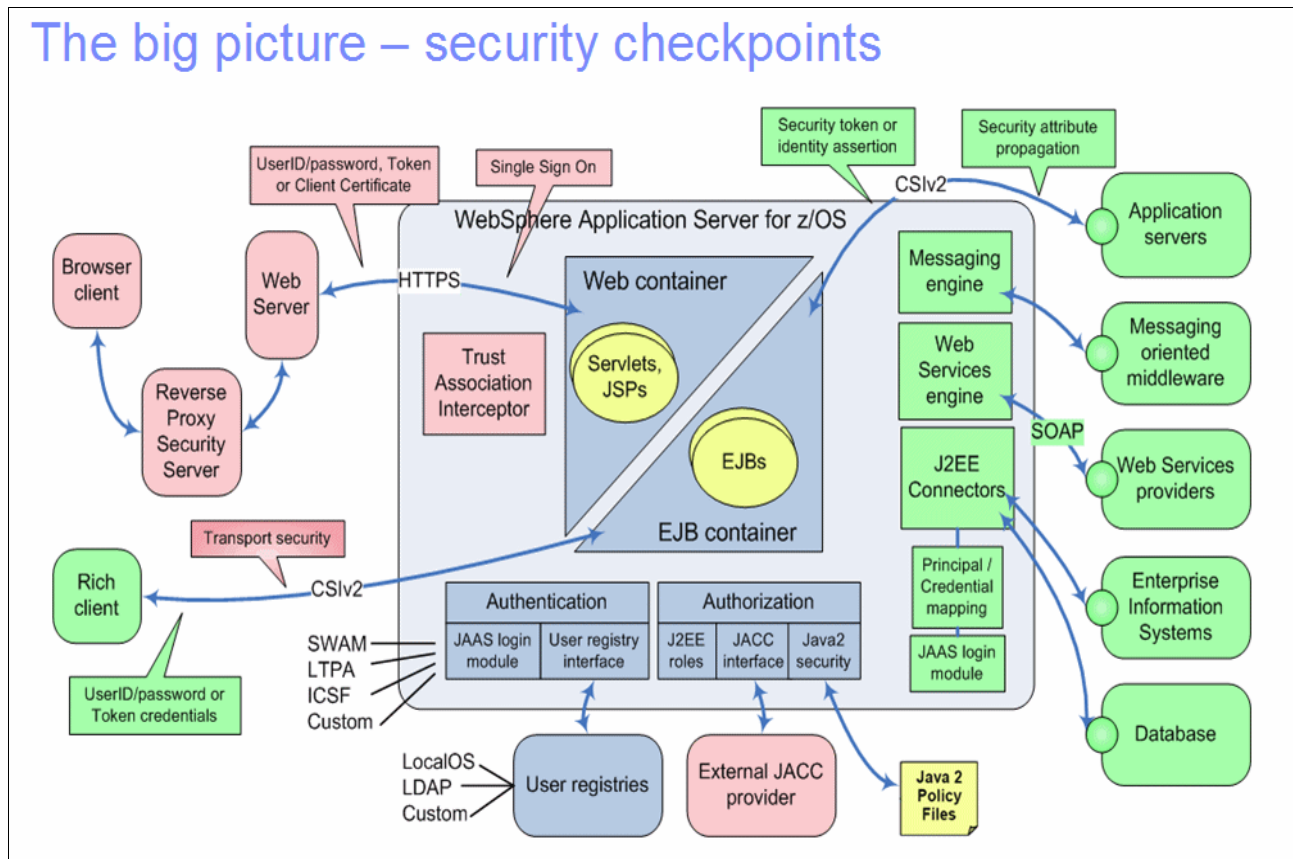


Figure 5-3 The big picture - security checkpoints

In the next section, we examine the WebSphere Application Server for z/OS Web container support in more detail.

Note: ICSF authentication mechanism was removed. SWAM was deprecated.

5.4.1 Web container security

Figure 5-4 presents the functions and interfaces to the Web Container as it applies to security. It handles requests for servlets (objects that receive a request and generates a response based on that request), JSP (Java Server Pages) files, and other components running on the server side. The Web container creates servlet instances, loads and unloads servlets, creates and manages requests and response objects, and performs other servlet management tasks.

The Web container processes servlets, JSPs, and other types of server-side includes. Requests are sent by the Web client. Web clients use the HTTP or HTTPS protocol to send the authentication information. Web authentication is required for Web clients when they access protected resources and is performed by the Web authentication module.

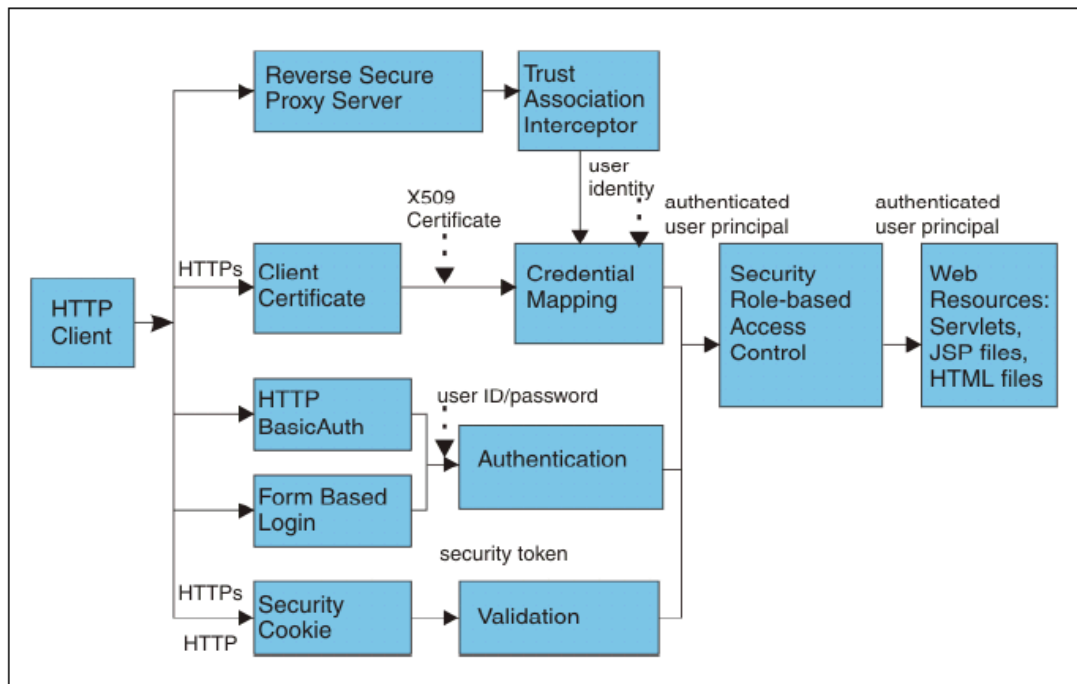


Figure 5-4 Web container security

The authentication method defines how the Web container authenticates the user. Before any authorization constraint is applied, the user must pass the authentication process using a configured mechanism. Some options for input into the authentication process referenced in Figure 5-4 are:

- ▶ Reverse secure proxy server: A proxy server that is installed within the neighborhood of one or more servers. A reverse proxy dispatches in-bound network traffic to a set of servers, presenting a single interface to the caller. You can use a reverse proxy to load balance a cluster of Web servers. In contrast, a forward proxy acts as a proxy for out-bound traffic.

In our discussion, the proxy server provides an additional layer of defense by separating or masquerading the type of server that is behind the reverse proxy.

- ▶ Client certificate authentication: The client is challenged for a valid and approved x.509 certificate (a standard for a public key infrastructure for single sign-on) to be able to start a communication with the server. After the server accepts the certificate, an encrypted channel is created between the client and the server.

- ▶ Basic authentication: The Web server sends a request to the client. The request contains the realm name in which the user is authenticated. The browser prompts the user for a user ID and password, which are encoded and included in every HTTP request.
- ▶ Form-based authentication: The application programmer provides a customized HTML form in which the user enters authentication data. The values that the end user supplies in the form are transmitted in clear text as parameter values in the first HTTP request. After the server authenticates the user, only an encrypted token is flowing between the server and the browser. This encrypted token does not contain the user password.
- ▶ Security cookie: When a single sign-on (SSO) is enabled, a security cookie is issued to an authenticated client, which can represent the user within the specified security domain.

5.5 The value of Tivoli Access Manager WebSEAL

In a Customer environment, the first authentication challenge takes place when the user accesses the portal.

IBM Tivoli® Access Manager WebSEAL component is the resource manager that manages and protects Web-based information and resources. As a reverse proxy, more complex authentication mechanisms can be enabled in the future without modifying the Web applications because the authentication logic is in the Tivoli Access Manager WebSEAL layer. Figure 5-5 positions WebSeal with respect to WebSphere Application Server security.

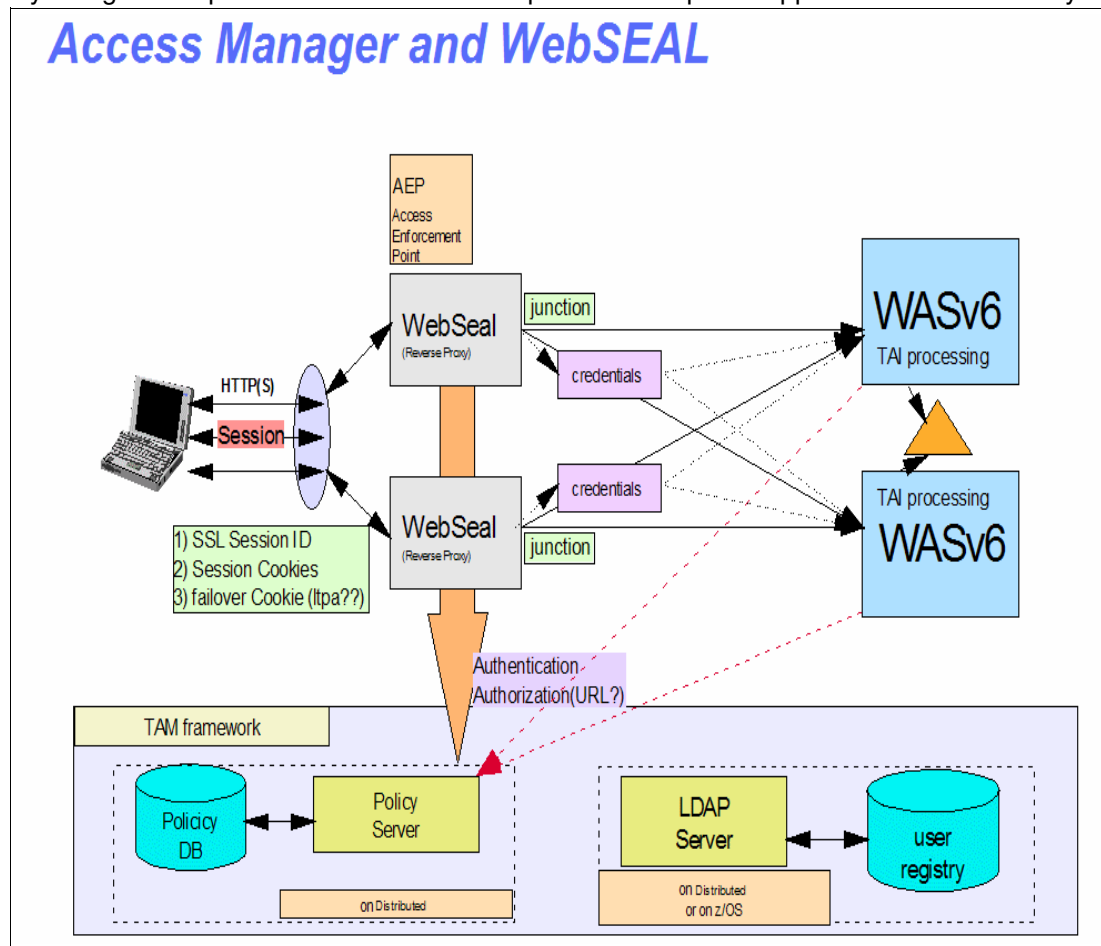


Figure 5-5 Access Manager and WebSEAL

There is a flow of control for the Tivoli Manager authorization process:

1. An authenticated client request for a resource is directed to the resource manager and intercepted by the policy enforcer process. The resource manager can be WebSEAL (for HTTP, HTTPS access) or a third-party application.
2. The policy enforcer process uses the Tivoli Access Manager authorization API to call the authorization service for an authorization decision.
3. The authorization service performs an authorization check on the resource, represented as an object in the protected object space.
4. The decision to accept or deny the request is returned as a recommendation to the resource manager (through the policy enforcer).
5. If the request is finally approved, the resource manager passes the request on to the application that is responsible for the resource, and the client receives the results of the requested operation.

We first discuss Open Transaction Manager Access (OTMA) security considerations and then continue with IMS Connect.

5.6 Open Transaction Manager Access security considerations

The /SECURE command controls the RACF security level for input from OTMA clients. It is used for administrative control of the IMS environment and as an emergency operations control command to throttle RACF activity, without requiring an IMS shutdown. The /SEC OTMA command is used with the CHECK, FULL, NONE, or PROFILE parameters. You can use the /DISPLAY OTMA command to show the security level that is currently in effect.

Use the OTMASE execution parameter to change the level of security desired at the IMS startup. The default is FULL. The /SECURE OTMA command overrides the value that you specify in the OTMASE keyword. If you do not specify the OTMASE keyword, IMS retains the OTMA security settings (which are established by the /SECURE OTMA command) after a warm start or emergency restart.

The parameters in the /SEC OTMA command mean:

► CHECK

Causes existing RACF calls to be made. IMS commands are checked using the RACF resource class of CIMS. IMS transactions are checked using TIMS.

► FULL

Causes the same processing as the CHECK parameter, but uses additional RACF calls to create the security environment for dependent regions.

► NONE

Does not call RACF within IMS for security verification.

► PROFILE

Causes the values in the security-data section of the OTMA message prefix for each transaction to be used.

If RACF is used for security, you must define the XCF group name and the member name (IMSXCF.group.member) in the FACILITY class. Note that the SAF interface is used, so if OTMA security is activated, you must define the FACILITY class.

5.6.1 Resume transaction pipe security

Resume transaction pipe (TPIPE) support adds security to asynchronous output messages that a Resume TPIPE retrieves. The support highlights are:

- ▶ New RIMS SAF/RACF security resource class:
 - Resume TPIPE Security is activated when RIMS is defined
 - It associates the TPIPE name with UserID/Group that can access it
 - Defines the TPIPE names as resources and PERMIT UserIDs/Groups
 - You can specify RCLASS=Rxxxxxxx on the Security Macro or in DFSDCxxx
- ▶ Requires OTMA Security of CHECK or FULL
- ▶ The OTMA Security User Exit Routine (DFSYRTUX) is used:
 - It is invoked after a call to SAF/RACF, regardless of the result
 - It is always called, if it exists, whether or not RIMS is defined
 - The default routine always provides RC=0 (allow access) for compatibility

5.6.2 Transaction member level security

In this section, we discuss the transaction member (TMEMBER) level security:

- ▶ New /SECURE OTMA TMEMBER command capability:
 - Allows each OTMA member to define its own security level
- ▶ NONE, CHECK, FULL, or PROFILE:
 - Can dynamically change the security level for a TMEMBER
 - Messages are processed with the security level that was in effect when they were received
- ▶ IMS command support:
 - /SECURE OTMA security-option TMEMBER member-name where security-option is NONE | CHECK | FULL | PROFILE
 - /DISPLAY OTMA and /DISPLAY TMEMBER member-name show TMEMBER level Security
 - OTMA security refresh by UserID
- ▶ OTMA caches RACF accessor environment elements (ACEEs) in memory for better performance when OTMA security is enabled:
 - Changes to RACF Profiles are not reflected in cached copies. OTMA provides “aging” for cached ACEEs
- ▶ Prior command support for refreshing cached ACEEs:
 - /SECURE OTMA REFRESH
 - Refreshes all ACEEs for all TMEMBERS
 - /SECURE OTMA REFRESH TMEMBER member-name
 - Refreshes all ACEEs for specified member-name

REFRESH invalidates cached ACEEs but they are not rebuilt until their next use. This spreads out the impact of refresh or OTMA security refresh by UserID.
- ▶ New command option to refresh a specific User Profile:
 - /SECURE OTMA REFRESH USER userID

Where userID is the User Profile for which ACEEs in all TMEMBERS are refreshed.

5.6.3 Accessor environment element Aging Value

OTMA caches RACF ACEEs in memory for better performance when OTMA security is enabled. Changes to RACF Profiles are not reflected in cached copies.

- ▶ OTMA provides “aging” for cached ACEEs as of IMS Version 10:
 - IMS Connect has new OAAV parameter on the DATASTORE statement of the HWSCFGxx member.
- ▶ Can specify the OTMA ACEE Aging Value in seconds:
 - IMS Connect passes this value during OTMA client-bid; 0 to 2147483647, with values 0 to 300 being reset to 300.
For example: DATASTORE=(ID=.....,OAAV=600,...). Default if not specified is 2147483647 (68 years).
- ▶ OTMA ACEE Aging Value can be displayed with IMS Connect commands VIEWHWS or VIEWDS.

Example 5-3 presents the output from the VIEWHWS.

Example 5-3 Aging value

```
DATASTORE=IMS1 STATUS=ACTIVE
GROUP=XCFGRP1 MEMBER=HWS2
TARGET MEMBER=IMS1
DEFAULT REROUTE NAME=HWS$DEF
RACF APPL NAME=APPLID1
OTMA ACEE AGING VALUE=600
```

5.6.4 IMS OTMA callout security

IMS transactions and commands that flow through OTMA from various clients are protected by current security classes, namely: IMS and CIMS. The responses are guaranteed to be delivered to the client that initiated the transactions and commands. However, output messages in the hold queue that are generated as a result of asynchronous processing are not protected by any security class. When those messages are retrieved by an OTMA client using RESUME TPIPE, a security exposure can occur.

The function provided by callout security protects these output messages by establishing a security class named RIMS within RACF or any non-IBM security product. Within this class, the security definitions are associated with the TPIPE name and with the list of user IDs or group names under this TPIPE.

The enhancement allows IMS installations to optionally authorize the user ID, together with the TPIPE name that is contained in the Resume TPIPE command message, to receive the output messages before any of these messages are sent to an OTMA client.

Implementation

IMS installations that want to take advantage of the callout security must:

- ▶ Define a new resource class, TPIPE name, and user IDs in RACF.
The resource class comprises the resource class type of “R” and the resource class name whose value is taken from the RCLASS parameter of the IMSCTRL macro. If RCLASS is omitted, the resource class name defaults to “IMS.” The resulting class then is “Rxxxxxxx”, where ‘xxxxxxx’ is the value of RCLASS, or “RIMS”, as the default.

- ▶ Ensure that the OTMA client includes:
 - The TPIPE name in the OTMA CTL prefix
 - The user ID in the security prefix header, in the message for the Resume TPIPE command.
- ▶ Code, assemble and link-edit the user exit in a library that is concatenated with IMS RESLIB under DD name STEPLIB or JOBLIB. The DFSYRTUX user exit is always called regardless of the success or failure in RACF authorization.

Authorization is performed when Resume TPIPE is initiated, but before retrieving the messages from the hold queue. The authorization procedure is comprised of two security levels.

The first security level is to determine if resource class “RIMS” or “Rxxxxxxx” is defined in the system. If it is not defined, the messages in the hold queue can be sent to the user through the OTMA client only after the user exit is invoked. If it is defined, authorization logic not only verifies and validates the security header, but also authorizes the user ID under the TPIPE name using RACF facilities.

The second security level is to invoke the user exit regardless of the success or failure of the first security level. The user exit can either take whatever results of the first security level, override its result, or add a more restrictive security rules.

When authorization is successful, output messages in the hold queue are returned to IMS Connect. A rejection message of the resume TPIPE command is sent to the client when authorization fails.

How this function is invoked or started

The user either defines a new resource class in RACF or codes the user exit to supplement the authorization of the user ID under a TPIPE name. At the minimum, the DFSYRTUX user exit is invoked even if the resource class, RIMS or Rxxxxxxx, is not defined in RACF.

Refer to *Security Server RACF Security Administrator's Guide*, SA22-7683 for guidelines about defining a resource class in RACF.

5.7 IMS Connect security considerations

The USERID=&userid parameter that is specified in the JOB card of the IMS Connect JCL is used as the security vehicle to ensure IMS Connect access to IMS. &userid must have READ access to IMSXCF.group.member. IMS OTMA provides security for the IMS XCF connection by defining and permitting IMSXCF.group.member in the RACF FACILITY class.

To configure security for the local option using RACF: You must add HWS.ICON_NAME as the SAF facility class name (whether you configured security with the IMS Connect configuration member or SETRACF command). ICON_NAME is how IMS Connect is defined in the ID parameter of the HWS statement in the IMS Connect configuration member. The resource that must access IMS Connect is the WebSphere Application Server, and UPDATE authority is required to update the RACF profile.

IMS Connect supplies some important security related features.

Client password change request

This facility was introduced with IMS Version 10 for an IMS Connect Client to request that a RACF password be changed. You can issue a RACF password change request to IMS Connect from your Java application using the HWSPWCH keyword in IMS TMRA Version 10. When IMS Connect is configured to perform RACF authentication, you can change RACF passwords of the userID specified in your message from your Java application by submitting a message that includes the password change request keyword HWSPWCH in the following format:

```
HWSPWCH old-password/new-password-1/new-password-2
```

The HWSPWCH keyword must appear at the beginning of the application data section of the message and be followed by one or more whitespace characters, the current password, a forward slash character, the new password, another forward slash, and then the new password again.

HWSPWCH is a defined keyword in user message exits: HWSSMPL0, HWSSMPL1, and HWSJava0. Exits must include the HWSPWCH0 object deck

The IMS TMRA format for this call is:

```
LLLL|IRM|OTMA|LLZZ HWSPWCH old-pswd/new-pswd1/new-pswd2|EOM
```

For RYO Clients, the format is:

```
LLLL|IRM|LLZZ HWSPWCH old-pswd/new-pswd1/new-pswd2|EOM
```

RACF mixed case password support

The support for RACF mixed case password in IMS Connect is aligned with the IMS V10 support for mixed case passwords. The capability in IMS Connect allows the password to be preserved exactly as the remote client provided and pass the string to RACF without translation to upper case.

RACF begins supporting the upper/lower case passwords in z/OS 1.7. This mixed case support is initially disabled, and you must enable this support through the RACF SETROPTS(MIXEDCASE) command.

This RACF enablement of this support does not constitute automatic IMS Connect usage of this support. You must also specify the usage of this support through a parameter in the configuration member or through IMS Connect commands:

- ▶ A new parameter, PSWDMC= in the HWS= statement, defines the option of mixed case passwords where PSWDMC=N is the default. PSWDMC parameter settings on HWS statement in HWSCFGxx member are HWS=(ID=...,RACF=Y,PSWDMC=YIN,...)
- ▶ IMS Connect also provides a command, SETPWMC, that can override the HWSCFGxx specification. It is specified as SETPWMC ON | OFF
- ▶ The PSWDMC keyword is also available to the IMS Connect UPDATE command as another way to request the support:

```
F imskonproc, UPDATE MEMBER TYPE(IMSCON)SET(PSWDMC(ONIOFF))
```

Datastore access control

Using datastore access control you can control access to Datastores by RACF userIDs. Part of the RACF Passticket support is set through the APPL parameter on the DATASTORE statement in HWSCFGxx member.

IMS Connect requires RACF=Y for password validation through:

```
RACROUTE REQUEST=VERIFY,APPL=RACF_APPL,... (APPL Class is used in Passticket processing)
```

RACF also verifies the user's authority to access the application, which saves the application from having to do a separate RACROUTE REQUEST=AUTH.

So even if you are not using Passtickets, you can use the APPL Class to control access to the datastore.

Override the DATASTORE APPL: IMS Connect users can override the DATASTORE APPL by setting IRM_APPL_NM. If this is a concern, you must comment out the override in the User Message Exits:

```
MVC OMUSR_APPL_NM,IRM_APPL_NM
```

5.7.1 IMS Connect SSL connections

To use Secure Sockets Layer (SSL) support with IMS Connect, you can either configure the IMS Connect support for SSL by using the IMS Connect SSL interface or by using the IBM z/OS Communications Server Application Transparent Transport Layer Security (AT-TLS) feature.

SSL and TLS protect the privacy and integrity of data that is transferred through a network. SSL rests on top of TCP/IP to provide a mechanism for secure sockets. SSL uses a combination of public and private keys and symmetric key encryption to authorize clients and servers to one another.

If you use AT-TLS, the use of SSL is transparent to IMS Connect and you do not need to configure the IMS Connect SSL interface. Your z/OS TCP/IP administrator configures and administers SSL. For information about configuring AT-TLS, see *z/OS V1R9.0 Communications Server IP Configuration Guide*, SC31-8775-11.

In addition to simplifying IMS Connect security implementation, AT-TLS provides greater flexibility with respect to the use of ports. IMS Connect support for SSL is restricted to only a single port. AT-TLS does not have this restriction.

Note: Transport Layer Security, Version 1 (TLS V1) is the successor to SSL 3.0 protocol. The IMS TMRA only supports TLS V1. There are no backward compatibility issues.

Setting up IMS Connect SSL ports

As an optional task, you can set up SSL (secure socket layer) by:

- ▶ On your z/OS system, verify that the SSL ports are active using the MVS Modify Command, Control Center, or MVS WTOR, for example, you can use the VIEWHWS command to display a list of the SSL ports and details their status. Ensure that some of the ports have the suffix *s* and that they are in the ACTIVE state.
- ▶ Refer to the “*IMS Connect SSL connections*” topic in the reference: *IMS V10 Communications and Connections Guide*, SC18-9703 on how to set up the SSL configuration in IMS Connect.

Note: For SSL support, set GSK_PROTOCOL_TL SV1 to ON. For more information, see the *IMS V10 Communications and Connections Guide*, SC18-9703.

In the next section, we examine the security considerations for the key IMS SOA Integration suite functions and tools.

5.8 IMS TM Resource Adapter security considerations

The IMS TM Resource Adapter security credentials are performed through the Administrative console through the IMS connection factory properties.

5.8.1 IMS connection factory properties

You must configure the custom properties of an IMS TM resource adapter connection factory to match the characteristics of the target EIS (Enterprise Information System).

When you create an IMS service definition or define an IMS connection factory to WebSphere Application Server, you must provide values for certain properties of the connection between the IMS TM resource adapter and IMS Connect. These connection properties are:

- ▶ **HostName:** This value is mandatory for TCP/IP connections only. It is ignored for local option connections. You must replace the value that is mentioned with the IP address or host name of the machine on which the target IMS Connect is running.
- ▶ **PortNumber:** This value is mandatory for TCP/IP connections only. It is ignored for local option connections. You must replace the value of "0" with the number of a port that the target IMS Connect for TCP/IP connections uses. Multiple sockets can be open on a single TCP/IP port, and you can configure IMS Connect to use multiple ports for communications with the IMS TMRA and other clients.
- ▶ **CMODedicated:** This value is optional and it applies to TCP/IP connections only. The default value of FALSE indicates that the connection factory generates shareable persistent socket connections, and the IMS TM resource adapter generates a clientID to identify the socket connection. Commit mode 0 and commit mode 1 interactions can use these connections. A value of TRUE indicates that the connection factory generates dedicated persistent socket connections, which require user-specified clientIDs to identify the socket connections.
- ▶ **SSLEnabled:** This value is optional, and it applies to TCP/IP connections only. The default value is FALSE. A value of TRUE indicates that this connection factory will be used to create SSL socket connections to IMS Connect using the HostName and PortNumber that are specified in its connection properties. This PortNumber must be configured as an SSL port in the IMS Connect configuration. A value of FALSE indicates that SSL sockets will not be used for connecting to the port that is specified in the PortNumber property.
- ▶ **SSLKeyStoreName:** This value is optional, and it applies to TCP/IP connections only. You must set SSLEnabled to TRUE. SSLPrivate keys and their associated public key certificates are stored in password-protected databases called keystores.
- ▶ **SSLKeyStorePassword:** This value is optional. It applies to TCP/IP connections only, and SSLEnabled must be set to TRUE. Specify the password for the keystore.
- ▶ **SSLTrustStoreName:** This value is optional, and it applies to TCP/IP connections only. You must set SSLEnabled to TRUE. For non-z/OS platforms, specify the fully-qualified path name of your JKS truststore file. For z/OS, specify the JKS name or the RACF keyring of the truststore. The same format is used for the values of the SSLKeyStoreName and SSLTrustStoreName properties. See the description of the SSLKeyStoreName property for a discussion of this format.

A truststore file is a key database file (keystore) that is intended to contain public keys or certificates.

- ▶ **SSLTrustStorePassword:** This value is optional, it applies to TCP/IP connections only, and `SSLEnabled` must be set to `TRUE`. Specifies the password for the truststore.
- ▶ **SSLEncryptionType:** The value of the `SSLEncryptionType` property is case-insensitive and optional. It applies to TCP/IP connections only, and `SSLEnabled` must be set to `TRUE`. Select the encryption type to `STRONG`, `WEAK`, or `ENULL`. `STRONG` and `WEAK` reflect the strength of the ciphers, which is related to the key length. When `ENULL` is specified, the IMS TM resource adapter uses a cipher spec whose name contains the string "NULL". Null encryption allows for authentication to take place during the SSL handshaking process, as is currently the case. After the handshaking process for a socket completes, which includes authentication, as required, all messages flow in the clear over that socket.
- ▶ **IMSConnectName:** This value is mandatory for local option connections only. Specify the job name of the target IMS Connect. If the `IMSConnectName` is specified, the Host name, Port number, and SSL-related properties are ignored. When an IMS Connect name is specified for a connection factory, that connection factory can only be used to create local option connections.
- ▶ **UserName:** This value is optional and is the default security authorization facility (SAF) user name that will be used for connections that this connection factory creates, if no `UserName` property is provided by the application component or the container.
- ▶ **Password:** This value is optional and is the password that will be used for connections that this connection factory creates, if the default user name is used.
- ▶ **GroupName:** This value is optional and is the IMS group name that will be used for all connections that this connection factory creates, if the default user name is used.

GroupName property: The `GroupName` property can only be provided in a component-managed EIS signon environment.

- ▶ **DataStoreName:** This value is mandatory and is the name of the target IMS datastore. You must replace the default value, "myDStrNm", with the ID parameter of the Datastore statement that is specified in the target IMS Connect configuration member. It also serves as the XCF member name for IMS during internal XCF communications between IMS Connect and IMS OTMA. The value that is specified for `DataStoreName` is case sensitive.
- ▶ **TraceLevel:** This value is optional and is the level of information to be traced.
- ▶ **TransactionResourceRegistration:** For IMS TMRA Version 9 only. This value is optional and is the type of transaction resource registration (transaction enlistment). This property was deprecated but is still supported for compatibility with older applications. Valid values are either `STATIC` (immediate) or `DYNAMIC` (deferred). If this property is set to `DYNAMIC`, the enlistment of the resource to the transaction scope is deferred until the resource is used for an interaction for the first time.

- ▶ MFS XMI Repository ID: For IMS TMRA Version 9 only. This value is optional and is a unique name for identifying the repository location. This ID applies to MFS transactions only and must match the repository field that is defined in the generated format handler of your application. The default for this field is "default".
- ▶ MFS XMI Repository URI: For IMS TMRA Version 9 only. This value is optional and specifies the physical location of the XMI repository. The URI valid formats for this field include:
 - file://path_to_xmi, where path_to_xmi is a directory on the local file system containing the xmi files, for example file://c:/xmi.
 - http://url_to_xmi, where url_to_xmi is a valid url that resolves to a directory containing the xmi files, for example http://sampleserver.com/xmi.
 - hfs://path_to_xmi, where path_to_xmi is the HFS directory on the host z/OS. This format is only supported for WebSphere Application Server for z/OS.

5.9 IMS SOAP Gateway security considerations

IMS TM SOAP Gateway uses IMS Connect and IMS OTMA to invoke an IMS transaction as a Web Service.

5.9.1 SOAP Gateway connection bundle

The connection bundle specifies the connection and security properties between IMS SOAP Gateway, IMS Connect, and IMS. The connection bundle properties refer to the location and other parameters that are required for creating a TCP/IP socket connection to IMS. You can create as many connection bundles as needed. However, each Web Service can be associated with only one connection bundle at a time, and you must specify the Web Service name in the Correlator file for that Web Service.

Each connection bundle consists of the following connection and security properties:

- ▶ Connection bundle name: A name to identify this set of connection properties.
- ▶ IMS Connect hostname: Specifies the name or IP address of the host system where IMS Connect is running.
- ▶ IMS Connect port number: The port number of IMS Connect.
- ▶ IMS Datastore: The name of the target IMS datastore. It must match the ID parameter of the Datastore statement that is specified in the IMS Connect configuration member. It also serves as the XCF member name for IMS during internal XCF communications between IMS Connect and IMS OTMA.
- ▶ User ID: The security authorization facility (SAF) user name that is used for the connection to IMS.
- ▶ Password: The SAF password that is used for the connection to IMS.
- ▶ Groupname: The SAF group name that is used for the connection to IMS.
- ▶ SSL Keystore name: Specifies the fully-qualified path name of the Keystore in which trusted certificates and private keys are stored.
- ▶ SSL Keystore password: Specifies the password for the keystore. The password length must be between 6 and 20 characters.
- ▶ SSL Truststore name: Specifies the fully-qualified path name of the Truststore in which trusted certificates are stored.

- ▶ **SSL Truststore password:** Specifies the password of the Truststore in which trusted certificates are stored. The password length must be between 6 and 20 characters.
- ▶ **SSL Encryption level:** Select the encryption type. A value of Strong indicates that a strong cipher suite must be used. A value of Weak indicates that a weak cipher suite must be used. A value of None indicates that no encryption is used. The None encryption level is used only for authentication.

5.10 Configuring the SOAP Gateway for SSL

To configuring the SOAP Gateway for SSL:

1. On a z/OS machine, set up the SSLPort and SLENVAR parameters for IMS Connect in the configuration member, as presented in Example 5-4. In this example, our member IM10IC1A resides in the IMS proclib data set.

Example 5-4 IMS Connect configuration member

```

HWS=(ID=IM10IC1A,XIBAREA=100,RACF=N,RRS=Y,SMEMBER=SM01)
TCPIP=(HOSTNAME=TCPIP,PORTID=(8999,28999),MAXSOC=2000,TIMEOUT=30000,
RACFID=PEGGYR,EXIT=(HWSSMPL0,HWSSMPL1,HWSOAP1,HWSIMS00),ECB=Y,
SSLPORT=(38991),SLENVAR=SSLPARMS)
DATASTORE=(ID=IM10,GROUP=IMSESA,MEMBER=IM10IC1A,TMEMBER=I10A)
ADAPTER=(XML=Y)

```

In the Proclib data set we require member SSLPARMS, as presented in Example 5-5.

Example 5-5 SLLPARMS member

```

GSK_PROTOCOL_SSLV2= GSK_PROTOCOL_SSLV2_ON
GSK_PROTOCOL_SSLV3= GSK_PROTOCOL_SSLV3_ON
GSK_PROTOCOL_TL SV1= GSK_PROTOCOL_TL SV1_ON
GSK_KEYRING_FILE= IMSConnKeyring
GSK_KEYRING_PW=
GSK_KEYRING_STASH_FILE=
GSK_KEYRING_LABEL = IMS Connect User Cert
GSK_CLIENT_AUTH_TYPE = GSK_CLIENT_AUTH_PASSTHRU_TYPE
GSK_SESSION_TYPE = GSK_SERVER_SESSION
GSK_V2_CIPHER_SPECS = 6321
GSK_V3_CIPHER_SPECS = 0906030201
DEBUG_SSL=OFF

```

2. Start IMS Connect.
3. From SDSF, select the display action (**DA**) option, and as presented in Figure 5-6, locate the IMS Connect that you just started.
4. Determine the owner that is associated with the proc.

Display Filter View Print Options Help											

SDSF DA G631 STLABB1 PAG 0 CPU/L 0/***							LINE 1-1 (1)				
COMMAND INPUT ==>							SCROLL ==> CSR				
NP	JOBNAME	StepName	ProcStep	JobID	Owner	C	Pos	DP	Real	Paging	SIO
	IM10IC1A	IM10IC1A	STEP1	STC01060	IMSCONN	NS	EC	3923	0.00	0.00	

Figure 5-6 Output from the SDSF DA option

- From TSO option 6, issue the RACDCERT command. Place the owner name that is located through the SDSF DA option output into the ID:

```
RACDCERT ID(IMSCONN) LISTRING(*)
```

After the command is complete, the output looks similar to Figure 5-7.

```

Digital ring information for user IMSCDNN:

Ring:
>IMSConnKeyring<
Certificate Label Name          Cert Owner      USAGE          DEFAULT
-----
IMS Connect Certauth           CERTAUTH       CERTAUTH       NO
WebSphereCA.P1CELL             CERTAUTH       CERTAUTH       NO
IMS Connect User Cert          ID(IMSCONN)    PERSONAL       YES

:~:~:~

```

Figure 5-7 RACDCERT digital ring information

The Certificate label of interest is the CERTAUTH for IMS Connect.

- Create a dataset to export the certificate using the following TSO command:

```
RACDCERT CERTAUTH EXPORT(LABEL('IMS Connect Certauth')) DSN('IMSCONN.CERTBIN')
FORMAT(CERTDER)
```
- From TSO, copy file to the HFS system using the following command:

```
OPUT 'IMSCONN.CERTBIN' '/u/jksingh/imsconn.cer' binary convert(no)
```
- Use the KEYTOOL GUI tool. Figure 5-8 shows the window capture using the tool.

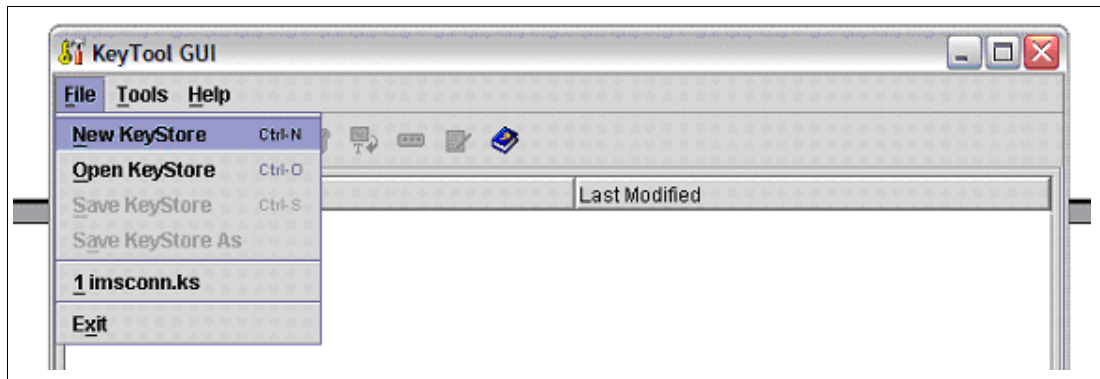


Figure 5-8 KeyTool GUI

- Click **Tools** → **Import Trusted Certificate**, as displayed on Figure 5-9 on page 103.

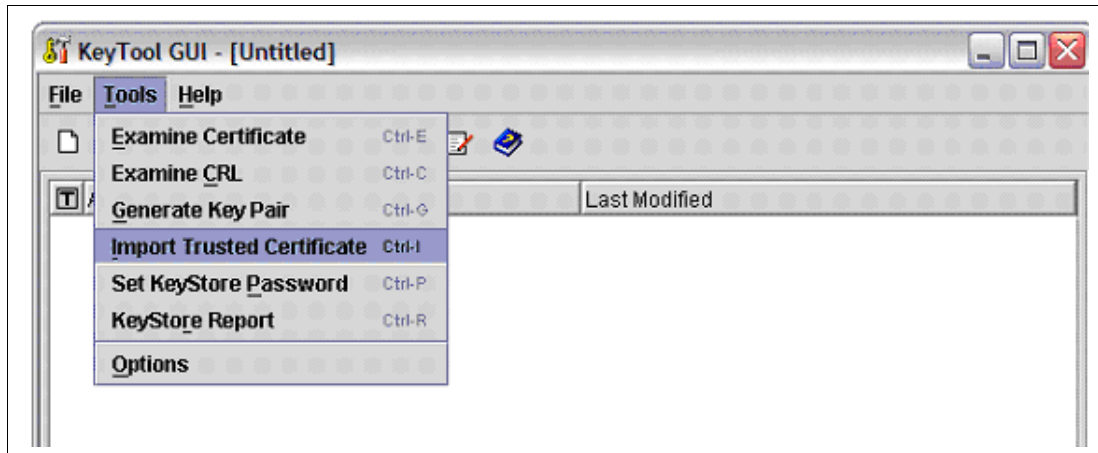


Figure 5-9 Import Trusted Certificate

10. Browse to the IMS Connect certificate, and click **Import**.

After a few **OK** clicks, it shows you the Certificate details that you are currently importing. Figure 5-10 presents the certificate details.

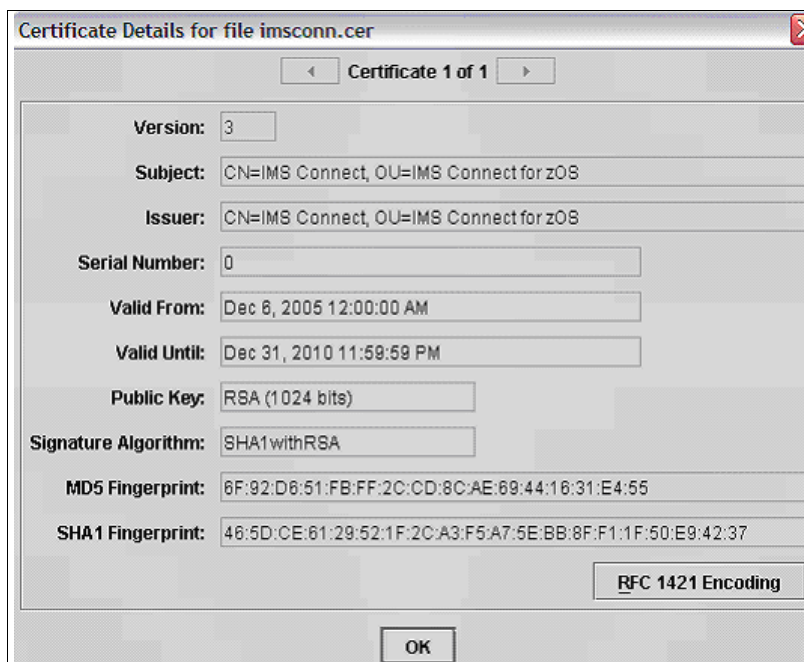


Figure 5-10 Certificate details

11. Select **OK** to the question if you want to accept it, click OK.

12. Enter an Alias and the KeyStore Password.

13. The Save Key Store As window is displayed, as shown in Figure 5-11 on page 104. Enter filename `imscconn.ks` (any name, but we use this for now), and click **Save**.

This is the name to update the SOAP bundle with.

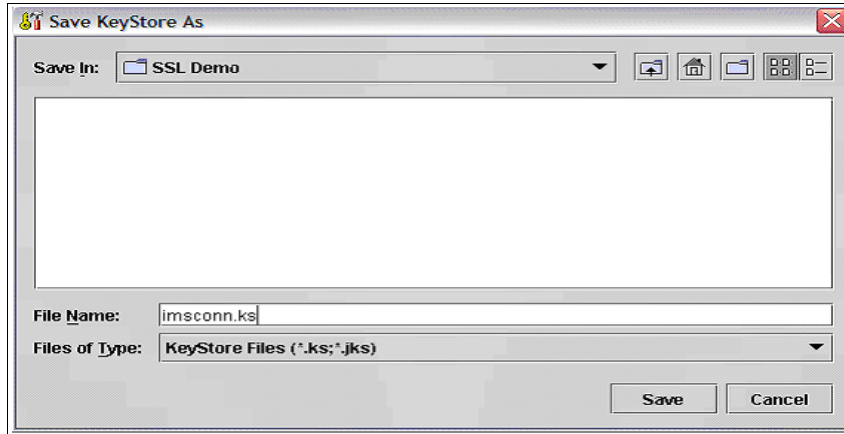


Figure 5-11 Save keystore as

14. Change the Keystore and Truststore names as follows:

```
<sslKeystoreName>C:\Program Files\IBM\IMS SOAP Gateway  
V9.2\server\webapps\imssoap\temp\client\imsconn.ks</sslKeystoreName>
```

```
<sslTruststoreName>C:\Program Files\IBM\IMS SOAP Gateway  
V9.2\server\webapps\imssoap\temp\client\imsconn.ks</sslTruststoreName>
```

At this stage, you set up IMS SOAP Gateway and IMS Connect with the SSL keys.

This is the conclusion of our discussion regarding security with respect to the IMS SOA features and tools.



Problem management for IMS services

In this chapter, we describe problem management methodologies and tooling in a SOA world compared with the best practices of the *Information Technology Infrastructure Library* (ITIL®). ITIL is a set of concepts and techniques for managing information technology infrastructure, development, and operations that is widely followed in z/OS installations.

OTMA, IMS Connect, and IMS Connect Extensions all provide diagnostic services, which are covered in the IBM Redbooks publication *IMS Connectivity in an On Demand Environment: A Practical Guide to IMS Connectivity*, SG24-6794.

We focus on those tools that assist you in your management of the reduction of defects, be they related to coding, process capture, architecture design, and the agility characteristics that are applied to application modernization projects.

6.1 Life cycle solutions to ensure functional quality

Software application quality depends on a great deal more than simply the absence of defects. Although the following are not necessarily the result of bugs, they all can badly affect user satisfaction:

- ▶ Improperly captured business processes
- ▶ Poorly extensible architecture
- ▶ Inflexible code

What can IT do to better ensure the high quality delivery of business applications? One solution comes from the utilization of an *Information life cycle management* (ILM) process. It is a methodology for managing information through its life cycle, from conception until disposal.

6.2 IBM IT life cycle management tools

There are tools that are available from IBM that analysts and developers can use to find and fix defects and design flaws early in the software development life cycle. Table 6-1 contains the tools that IBM offers in this area to help manage the IT life cycle.

Table 6-1 IT life cycle management tools

IBM Rational ClearQuest®	Flexible workflow management, defect, and change tracking across the project life cycle.
IBM Rational RequisitePro®	A requirements and use case management tool that promotes better communication, enhances teamwork, and reduces project risk.
IBM Rational Team Unifying Platform	Provides teams with the infrastructure tools, processes, and integrations that they need to work together more effectively. This solution unifies a team by providing common access to development assets, communication alerts, and processes to maintain project momentum and focus.
IBM Rational Software Architect	A design and construction tool for software architects and senior developers who are creating applications for the Java platform or in C++ that leverages model-driven development with the UML and unifies all aspects of software application architecture.
IBM Rational Functional Tester	An advanced, automated functional and regression testing tool for testers and GUI developers who need superior control for testing Java, VS.NET, and Web-based applications.
IBM Tivoli Composite Application Manager for Response Time Tracking	Monitors and traces application transactions end-to-end to identify and isolate end user response time performance problems.
IBM Tivoli Monitoring	Automates the monitoring of essential system resources to detect bottlenecks and potential problems and to automatically recover from critical situations.
IBM Tivoli OMEGAMON DE for Distributed Systems	Collates information from a wide variety of systems management monitors in a single location, which includes multiple OMEGAMON XE monitors and third-party software.

There are two tools of special interest for problem management:

- ▶ IBM Rational ClearQuest
- ▶ IBM Rational Functional Tester

6.2.1 Rational ClearQuest

SOA implementations are potentially more complex than traditional IT projects. Because SOA projects can contain more products and versions, team members, and sites that are involved in software delivery, greater risk of software quality issues and project delays can occur. Lack of communication, coordination, and prioritization across workgroups can affect overall effectiveness. Disconnected functional teams can generate redundant or inconsistent information, and compliance with internal and external mandates becomes even more difficult to manage.

By automating interactions among disparate project functions and groups, using Rational ClearQuest software you can eliminate software errors that occur because of manual processes and hand-offs, mis-communication, and inconsistent information. Built-in defect and change tracking capabilities prevent skipped steps or incomplete activities. Streamlined processes can lead to faster and more frequent test cycles, which allows you to detect errors earlier in the software delivery cycle.

Figure 6-1 displays the results of a query of high-severity defects that are reported by Rational ClearQuest.

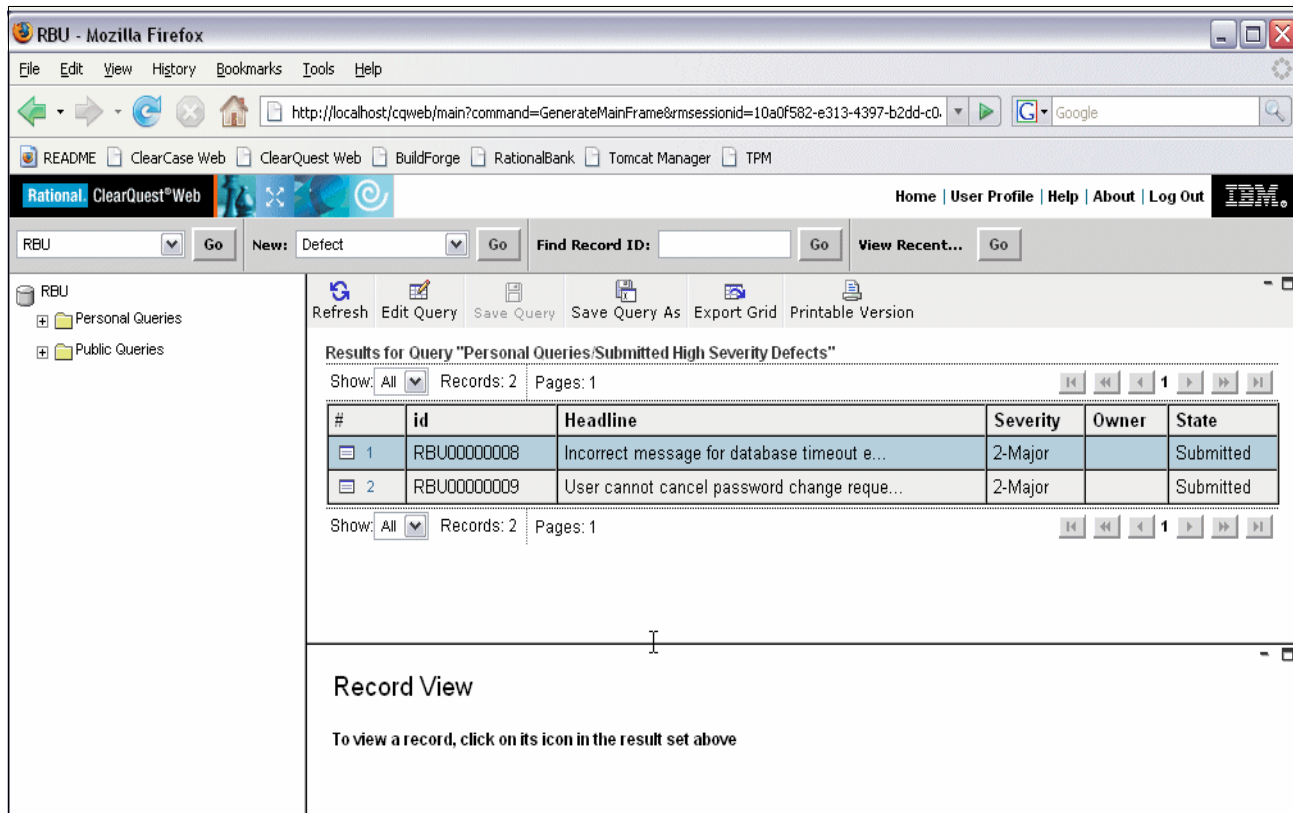


Figure 6-1 Results of query of high severity defects reported by Rational ClearQuest

For more information, refer to the Web site:

<http://www.ibm.com/software/awdtools/clearquest/index.html>

6.2.2 Rational Functional Tester

Using IBM Rational Functional Tester software you can automate functional and regression testing. It provides automated testing of Java, Web-based, and .NET Windows Forms applications on Microsoft Windows and Linux platforms.

The major features are:

- ▶ Ensures playback resilient to application changes with ScriptAssure® technology
Frequent changes to an application's user interface can hamper test script execution. ScriptAssure technology is utilized to accommodate these changes and avoid increases in maintenance overhead. ScriptAssure uses fuzzy matching algorithms to locate objects during test execution, even if the objects were changed since test creation.
- ▶ Increases script re-use with wizards for data-driven test creation
Data driven tests are functional tests that perform the same series of test actions but with varying data. IBM Rational Functional Tester can automatically detect data that is entered during test recording and prepare the test for data-driven testing. Using a spreadsheet-like data editor, you can then create customized data sets to be used by the test during playback. In this way, you can achieve test script re-use without time consuming manual coding.
- ▶ Streamline™ automation with keyword testing
Using the Rational Functional Tester you can define and automate keywords that can be reused in both functional and manual tests, which promotes script reuse and enables manual testers to easily and selectively leverage the power of automation within manual test cycles.

6.3 Omegamon XE and IMS problem management

Omegamon XE for IMS on z/OS is used to ensure that your IMS systems are running at peak performance, which means that you must know when the first signs of trouble enter your environment. Omegamon XE for IMS on z/OS provides alerts and information in the following SOA-related areas:

- ▶ IMS Connect throughput and response time summarized by transaction, data store, port, client ID, and user ID.
- ▶ IMS Connect TCP/IP usage summaries
- ▶ Resume TPIPE usage summaries
- ▶ IMS Connect exception events
Using the IMS Connect workspaces you can monitor TCP/IP transaction requests as IMS Connect is processing them, which provides a better understanding of the activity within the IMS Connect system, such as response times, resource availability, and data throughput.
- ▶ Version 4.1.0 provides increased precision for *Transaction Reporting Facility* (TRF) response time components, which includes input/output queue time, processing time, internal response time, and total response time
- ▶ An intuitive Web browser interface that shares the look and feel across many Tivoli OMEGAMON interfaces, virtually eliminates the need for special training.

For more information about Omegamon XE for IMS on z/OS, refer to the Web site:

<http://www.ibm.com/software/tivoli/products/omegamon-xe-ims/>

6.4 Problem determination tooling

In this section, we provide an overview of the available problem determination tools.

6.4.1 The IBM problem determination tools for System z

To effectively build and service applications, developers need robust, easy-to-use tools to compile, test, and debug their applications. The IBM z/OS Problem Determination and Deployment Tools comprise a suite of tools for improving the health of a company's application portfolio and at the same time address the growing need for cost-effective application development and testing. These tools are:

- ▶ Fault Analyzer (FA)
- ▶ File Manager (FM)
- ▶ Application Performance Analyzer
- ▶ Workload Simulator (WSim)
- ▶ Debug Tool (DT) and Debug Tool Utilities and Advanced Functions (DTU&AF)

Figure 6-2 shows the Problem Determination tools for System z.

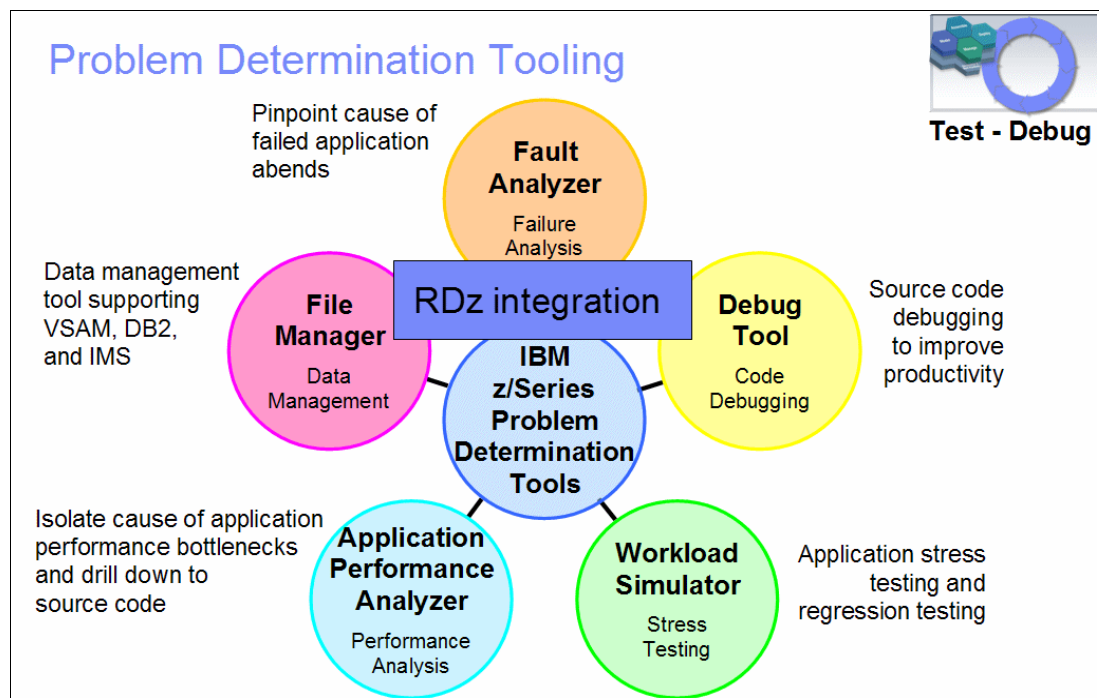


Figure 6-2 Problem determination tooling

By integrating the tools in Figure 6-2 with IBM Rational Developer for System z, application developers can perform tasks from their development environment.

We introduce these tools to you in the following sections.

Fault Analyzer for z/OS

IBM Fault Analyzer for z/OS gathers information about an application and the surrounding environment at the time of an abnormal end (abend), which provides you with valuable information that you need to develop and test your applications.

After analyzing information about your application and its environment, Fault Analyzer generates an analysis report that describes the problem in terms of application code, which means that you are not forced to interpret a low-level system dump or system-level error messages. The reason for the abend is made available to you sooner and with less effort.

For more information about the Fault Analyzer for z/OS, refer to the Web site:

<http://www.ibm.com/software/awdtools/faultanalyzer/>

File Manager for z/OS

The IMS component of File Manager (FM/IMS) is an ISPF application in which you can manipulate data that is stored in IMS databases. Using FM/IMS, you can:

- ▶ Display data from one or more IMS segment occurrences and see their relationship to other segment types within the database.
- ▶ Edit and update data in IMS segment occurrences.
- ▶ Insert segment occurrences into an IMS database.
- ▶ Delete segment occurrences from an IMS database.
- ▶ Extract a subset of IMS data to a flat file.
- ▶ Load data into IMS databases.
- ▶ Print selected data or entire databases.

For more information about the File Manager for z/OS, refer to the Web site:

<http://www.ibm.com/software/awdtools/filemanager/>

Application Performance Analyzer for z/OS

Application Performance Analyzer for z/OS measures and reports how your applications use resources in a z/OS environment, which provides the information that you need to isolate performance problems in applications and to test the effect of increased workloads on your systems.

It monitors, analyzes, and reports on the performance of CICS, Assembler, COBOL, PL/I, C/C++, DB2, IMS, and WebSphere MQ applications. It also integrates with the Fault Analyzer for z/OS and the Debug Tool for z/OS.

For more information about the Application Performance Analyzer for z/OS, refer to the Web site:

<http://www.ibm.com/software/awdtools/apa/>

Workload Simulator for z/OS

IBM Workload Simulator for z/OS and OS/390 can simulate a network of terminals and its associated messages. It is ideal for stress, performance, regression, function, and capacity planning tests. It provides support for SNA, CPI-C (LU 6.2), TCP/IP, Telnet 3270, 3270E and 5250 clients, Telnet line mode network virtual terminal clients, simple TCP and UDP clients, FTP clients, and multiple client applications that run on top of TCP/IP.

For more information about the Workload Simulator for z/OS and OS/390, refer to the Web site:

<http://www.ibm.com/software/awdtools/workloadsimulator/>

Debug Tool overview

Using the IBM Debug Tool for z/OS you can examine, monitor, and control the execution of C, C++, COBOL, and PL/I programs. You can use it with the GUI that is provided in IBM Rational Developer for System z, WebSphere Developer for System z, or WebSphere Developer Debugger for System z. It supports batch, TSO, CICS, DB2, DB2 stored procedures, IMS, and UNIX System Services.

Debug Tool interfaces

You can use the Debug Tool to debug your programs in batch mode, interactively in full-screen mode, or in remote debug mode. These terms identify the types of debugging interfaces that the Debug Tool provides:

- ▶ Batch mode: You can use the Debug Tool command files to predefine a series of Debug Tool commands to be performed on a running batch application. The results of the debugging session are saved to a log, which you can review at a later time.
- ▶ Full-screen mode: The Debug Tool provides an interactive full-screen interface on a 3270 device with debugging information that is displayed in three windows:
 - A Source window in which to view your program source or listing
 - A Log window that records commands and other interactions between the Debug Tool and your program
 - A Monitor window in which to monitor changes in your program

You can debug an IMS transaction that is running on an IMS MPP region in full-screen mode through a VTAM® terminal facility.

- ▶ Remote debug mode: In remote debug mode, the host application starts the Debug Tool, which uses a TCP/IP connection to communicate with a remote debugger on your Windows workstation.

Using the Debug Tool, in conjunction with a remote debugger, you can debug host programs, which includes batch programs, through a graphical user interface on the workstation. The following remote debuggers are available:

- ▶ WebSphere Developer Debugger for System z
- ▶ Compiled Language Debugger component of Rational Developer for System z
- ▶ Compiled Language Debugger component of WebSphere Developer for System z
- ▶ Compiled Language Debugger component of WebSphere Studio Enterprise Developer

For more information about the Debug Tool for z/OS, refer to the Web site:

<http://www.ibm.com/software/awdtools/debugtool/>

For more information about the Problem Determination Tools for System z, refer to the Reference Guide that is available from:

<ftp://ftp.software.ibm.com/software/htp/pdtools/PD-Tools-Reference-Guide-Sept-2007-WS011258-USEN-01.pdf>

Also, visit this Web site for information about the Problem Determination Tools for System z:

<http://www.ibm.com/software/awdtools/deployment/>

6.5 IMS Connect tracing

Figure 6-3 on page 112 outlines the general message flow in and out of IMS Connect. The IMS Connect internal, in-core wrap-around trace creates entries with module entry and exit

points, plus error return codes. The trace table size and level of tracing (high, medium, or low) are set in the *base primitive environment* (BPE) proclib member pointed to through parameter BFECFG.

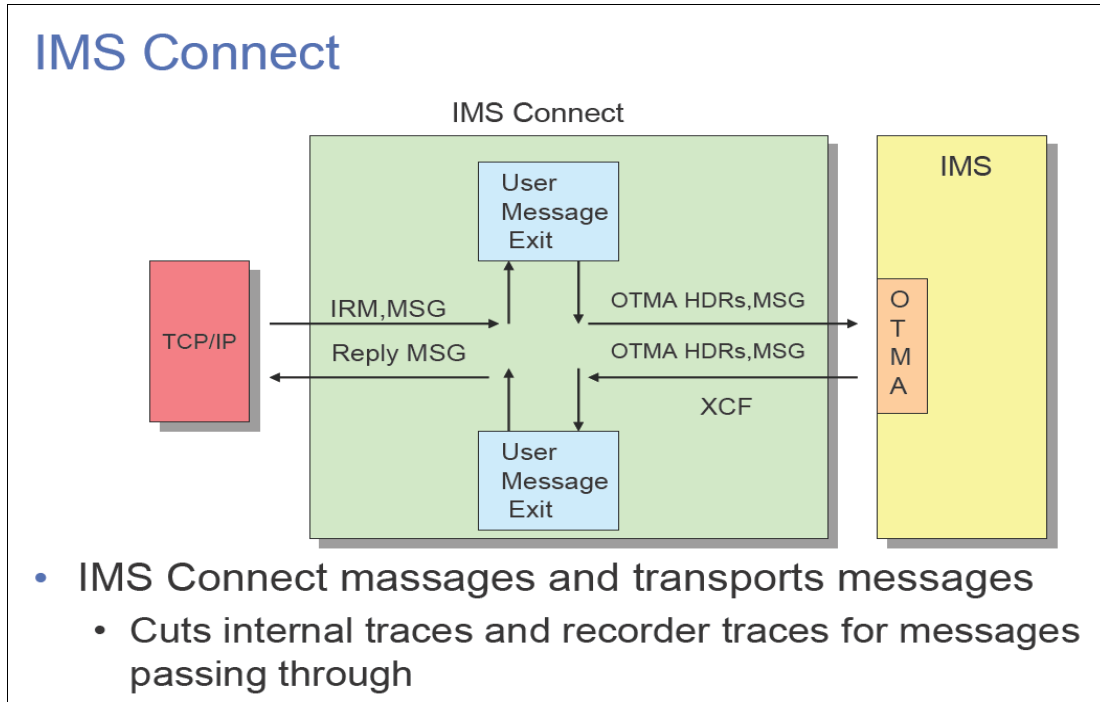


Figure 6-3 IMS Connect message flows

Another trace facility that is available is the IMS Connect recorder trace, which records messages that pass through IMS Connect, such as:

- ▶ Messages received from clients
- ▶ Messages sent to IMS/OTMA
- ▶ Messages received from IMS/OTMA
- ▶ Messages sent to clients

It records to an external data set using a HWSRCORD DD statement that is placed in the IMS Connect start-up member. The data set does not wrap around and there are no spares, so when the data set is full, the trace is disabled. No message is issued, but an internal BPE Trace entry is created, as shown here:

```
ENVV :RCDR0 RECORD DATA 00890000 C4C1E3C1 E2C5E340 C6E4D3D3
00000000DATA SET FULL
```

The trace is started and stopped with RECORDER OPEN and RECORDER CLOSE commands.

6.6 IMS TM Resource Adapter logging and tracing

In the *IMS TM Resource Adapter User's Guide and Reference*, SC19-1211-02, there is a diagnosing problems section that provides guidance about:

- ▶ Diagnosing problems accessing IMS from Java applications
- ▶ Diagnosing problems with asynchronous callout requests
- ▶ Exceptions that involve output messages

- ▶ Logging and tracing with the IMS TMRA adapter
- ▶ IMS TMRA messages and exceptions

Figure 6-4 describes the manner that you can set up for both logging and tracing for IMS TMRA.

Logging and tracing setup

- **Set value of TraceLevel property in J2C Connection Factory's Custom properties:**
 - ▶ 0 - RAS_TRACE_OFF
 - ▶ 1 - RAS_TRACE_ERROR_EXCEPTION (default)
 - ▶ 2 - RAS_TRACE_ENTRY_EXIT (method entry and exit)
 - ▶ 3 - RAS_TRACE_INTERNAL (includes buffer tracing)

- **Change Log Detail Levels in WAS:**
 - Run administrative console
 - Go to Troubleshooting, Logs and Trace
 - Click 'Change Log Detail Levels'
 - Set the log detail string to:
 - *=info: com.ibm.ejs.j2c.*=all: com.ibm.connector2.*=all: WAS.j2c=all
 - Save and restart WAS

- **Log can be found in the following or similar location:**
 - C:\Program Files\IBM\<WAS Install Dir>\runtimes\base_y61\profiles\<AppSrv01>\logs\server1

Figure 6-4 IMS TMRA logging and tracing setup

6.7 SOAP Gateway tracing

IMS SOAP Gateway User's Guide and Reference Version 10 Release 1, SC19-1290-01 contains a troubleshooting section that guides you through diagnosing runtime errors and lists the messages for IMS SOAP Gateway. To set the trace level for SOAP Gateway:

1. Start the IMS SOAP Gateway Deployment utility.
2. Select **Task 4: Update IMS SOAP Gateway properties**
3. Specify the trace level by using one of the the following values:
 - 1.Fatal(default)
 - 2.Error
 - 3.Warn
 - 4.Information
 - 5.Debug
4. Restart IMS SOAP Gateway.

6.8 TCP/IP problem analysis

TCP/IP protocol itself is robust, and often problems are due to mishandling by the client/server software, hardware, or are bandwidth related.

There are facilities that can assist when analyzing TCP/IP situations:

- ▶ Use the PING facility.
- ▶ Traces cut on either end of a TCP/IP connection, such as the IMS Connect recorder trace or the IMS TM RA and SOAP Gateway traces, often contain sufficient data to assist in *problem source identification* (PSI).
- ▶ Deploy network sniffers to trace packets and to collect data from other utilities, such as ipconfig and netstat.

Netstat displays statistics for IP, TCP, UDP, & ICMP protocols. It is useful in diagnosing and monitoring the network. It can be entered from:

- TSO
- MVS Console through Display TCP/IP command
- Unix System Services shell environment
- A distributed platform

For more information, refer to:

z/OS V1R6.0 Communications Server: IP System Administrator's Commands,
SC31-8781-04

6.9 Logging and tracing on WebSphere Application Server

If problems arise after application deployment to WebSphere Application Server using the IMS TMRA, you can trace Java application flow.

The IMS TMRA, in addition to other Java EE components, provide controls for logging and tracing component information. A trace file is created when these controls are set for logging and tracing. When you run your Java application using WebSphere Application Server, trace entries are created.

Note: Ensure that only one client is running when the trace is on.

Refer to the logging and tracing section within the manual:

IMS TM Resource Adapter User's Guide and Reference, SC19-1211-02 for details about how to set up both WebSphere Application Server and TMRA to trace.

The output of the trace is located in the following or a similar location:

C:\Program Files\IBM\<WAS Install Dir>\runtimes\base_v61\profiles\AppSrv01\logs\server1

Figure 6-5 on page 115 shows an example of a send flow.

Send example:

```
[07/08/08 16:16:21:650 EST] 29bee468 IMSManagedCon d *** IMS Connector for Java V9.1.0.1.4 ***
[07/08/08 16:16:21:650 EST] 29bee468 IMSManagedCon d ->
[com.ibm.connector2.ims.ico.IMSTCPIPManagedConnection@4c906405.getConnection(Subject,ConnectionRequestInfo)]

[07/08/08 16:16:21:650 EST] 29bee468 IMSManagedCon d *** IMSConnectionRequestInfo: null ***

[07/08/08 16:16:21:650 EST] 29bee468 IMSManagedCon d <-
[com.ibm.connector2.ims.ico.IMSTCPIPManagedConnection@4c906405.getConnection(Subject,ConnectionRequestInfo)]
...
[com.ibm.connector2.ims.ico.IMSTCPIPManagedConnection@4c906405.call(IMSConnection, IMSInteractionSpec, Record, Record
UserName=ZSG509 Password=***** GroupName=]

[07/08/08 16:16:21:660 EST] 29bee468 IMSManagedCon d *** IMSInteractionSpec: imsRequestType=1 commitMode=1
interactionVerb=1 executionTimeout=10000 socketTimeout=15000 mapName= ltermName= ***

[07/08/08 16:16:21:660 EST] 29bee468 IMSManagedCon d <->
[com.ibm.connector2.ims.ico.IMSTCPIPManagedConnection@4c906405.interactionPending()]
...
[com.ibm.ims.ico.IMSTCPIPAdapter@4c9be405.isConnected()]

[07/08/08 16:16:21:660 EST] 29bee468 IMSManagedCon d <->
[com.ibm.connector2.ims.ico.IMSTCPIPManagedConnection@4c906405.buildInputMsg(byte [])]

[07/08/08 16:16:21:660 EST] 29bee468 IMSManagedCon d -> [com.ibm.ims.ico.IMSTCPIPAdapter@4c9be405.send(byte [])]

[07/08/08 16:16:21:660 EST] 29bee468 IMSManagedCon d Buffer sent:
[07/08/08 16:16:21:660 EST] 29bee468 IMSManagedCon d [
[07/08/08 16:16:21:660 EST] 29bee468 IMSManagedCon d 00000229 001c0100 5cc8e6e2 d1c1e55c |.....□.*HWSJAV*| : 16
[07/08/08 16:16:21:660 EST] 29bee468 IMSManagedCon d 00000000 c0310000 c8e6e2e3 f5f5d4e3 |....{...HWST55MT| : 32
[07/08/08 16:16:21:660 EST] 29bee468 IMSManagedCon d 01400000 00004040 40404040 4040a0f0 |. .... μ0| : 48
[07/08/08 16:16:21:660 EST] 29bee468 IMSManagedCon d 00000000 00000000 00000000 00010000 |.....| : 64
```

Figure 6-5 Trace of a send

Figure 6-6 on page 116 depicts the receive flow and an exception, where the RA error messages have IConnnnE format.

Receive example:

```
[07/08/08 16:16:21:700 EST] 2c4c6468 IMSManagedCon d <-
[com.ibm.ims.ico.IMSTCPIPAdapter@3775241a.receive (IMSInteractionSpec)]

[07/08/08 16:16:21:700 EST] 2c4c6468 IMSManagedCon d <-
[com.ibm.connector2.ims.ico.IMSTCPIPManagedConnection@3773a41a.receive()]

[07/08/08 16:16:21:710 EST] 2c4c6468 IMSManagedCon d Buffer received:
[07/08/08 16:16:21:710 EST] 2c4c6468 IMSManagedCon d [
[07/08/08 16:16:21:710 EST] 2c4c6468 IMSManagedCon d 00000872 5cc8e6e2 d1c1e55c 01800000 |...K*HWSJAV*.III.| : 16
[07/08/08 16:16:21:710 EST] 2c4c6468 IMSManagedCon d 0000f9f9 f9f94040 4040a0f0 00000f5b |..9999  u0...$| : 32
[07/08/08 16:16:21:710 EST] 2c4c6468 IMSManagedCon d 00000000 00000000 00010000 00481020 |.....3..| : 48
[07/08/08 16:16:21:710 EST] 2c4c6468 IMSManagedCon d 0010c9d8 e4d5c6d4 e340bfc5 81166a3a |..IQUNFMT 4EA.| : 64
```

Exception example (RA error messages have **ICOnnnnE** format):

```
07/08/08 10:48:59:191 CDT 00000067 ConnectionMan 3 The com.ibm.connector2.ims.ico.IMSTCPIPManagedConnection@4ebd771a could not be
|
07/08/08 10:48:59:191 CDT 00000067 ConnectionMan E J2CA0292E: The ManagedConnection from resource eis/ims/PUMOKA30_jyd38-d95/IMSDE
07/08/08 10:48:59:204 CDT 00000067 IMSTCPIPManag 3 javax.resource.ResourceException: IC00010E: lazyEnlist (ManagedConnection) error
07/08/08 10:48:59:204 CDT 00000067 IMSTCPIPManag 3 javax.resource.ResourceException: IC00010E: lazyEnlist (ManagedConnection) error
07/08/08 10:48:59:204 CDT 00000067 IMSTCPIPManag 3 at com.ibm.connector2.ims.ico.IMSManagedConnection.call (IMSManagedConnection.j
07/08/08 10:48:59:205 CDT 00000067 IMSTCPIPManag 3 at com.ibm.connector2.ims.ico.IMSConnection.call (IMSConnection.java:215)
07/08/08 10:48:59:205 CDT 00000067 IMSTCPIPManag 3 at com.ibm.connector2.ims.ico.IMSInteraction.execute (IMSInteraction.java:498)
07/08/08 10:48:59:205 CDT 00000067 IMSTCPIPManag 3 at statefarm.framework.jcaimsframeworkw0099456.BaseJcaDao.invokeTransaction (Ba
07/08/08 10:48:59:205 CDT 00000067 IMSTCPIPManag 3 at statefarm.domain.utility.domainutilityw0086529.dao.DomainBaseJcaDAO.invoke (
07/08/08 10:48:59:205 CDT 00000067 IMSTCPIPManag 3 at statefarm.domain.agreement.requestbrokerw0090110.dao.RequestBrokerDAO.invo
07/08/08 10:48:59:205 CDT 00000067 IMSTCPIPManag 3 at statefarm.domain.agreement.requestbrokerw0090110.so.RequestBrokerSO.invokeS
07/08/08 10:48:59:205 CDT 00000067 IMSTCPIPManag 3 at statefarm.framework.controllerw0089701.delegates.requestbroker.RequestBroke
07/08/08 10:48:59:206 CDT 00000067 IMSTCPIPManag 3 at statefarm.framework.controllerw0089701.delegates.requestbroker.RequestBroke
07/08/08 10:48:59:206 CDT 00000067 IMSTCPIPManag 3 at statefarm.framework.controllerw0089701.serviceprocessing.RequestDataProvide
07/08/08 10:48:59:206 CDT 00000067 IMSTCPIPManag 3 at statefarm.framework.controllerw0089701.serviceprocessing.DataProviderImpl.i
07/08/08 10:48:59:206 CDT 00000067 IMSTCPIPManag 3 at statefarm.framework.controllerw0089701.serviceprocessing.ServiceFacadeImpl.
```

Figure 6-6 Receive and exception trace example

This concludes our discussion about some of the tooling that can assist in problem determination for SOA development and deployment activities.



Performance considerations for IMS Services

For multi-user Web applications, reliable, efficient performance is a necessity, not a luxury. Even small performance deficiencies and scalability failures can bring business to a halt or cause customers to take their business elsewhere. To capture and correct performance problems before deployment, software development and test teams must proactively measure an application's ability to rapidly and accurately support multiple concurrent users.

In this chapter, first we discuss the performance review process. Then, we examine the performance tuning elements that are associated with TCPIP, WebSphere Application Server, IMS Connect, IMS Connect Extensions, and the Language Environment®.

Finally, we introduce performance monitoring tooling that you can use in IMS Services environments.

7.1 Measuring and monitoring performance

There are established best practices that are associated with IT performance that can be applied in general to IMS services. In this section, we introduce these best practices.

7.1.1 Best general practices for measuring performance

The following are some key performance measurement best practices:

- ▶ Take your measurements during steady-state (do not include ramp-up/ramp-down times) processing periods.
- ▶ Only make one change at a time and repeat tests after making changes.
- ▶ Make sure your results are repeatable. Do not rely on data from a single run. You typically want to do at least three runs to make sure you are getting consistent results.
- ▶ Your runs need to be long enough to get repeatable results (typically 10-15 minutes). Investigate large variances between runs and try to keep the run-to-run variance below 4%.
- ▶ Test on an isolated network.

7.2 Performance Measurement Infrastructure

Performance Measurement Infrastructure (PMI) is a term that is associated with WebSphere Application Server that applies to transaction performance metrics. WebSphere Application Server lets you monitor runtime and application components through its Performance Monitoring Infrastructure (PMI) tooling. Some key elements are:

- ▶ System performance:
 - CPU
 - I/O
 - Memory (for paging, data space utilization, control block placement, in core tables and data stores, and buffering use)
- ▶ Application performance:
 - Servlet/EJB requests
 - Servlet/EJB response time
 - Live HTTP sessions

The performance data from the PMI client is organized into modules. In WebSphere Application Server, PMI provides performance data about the following runtime modules:

- ▶ Connection and database connection pools
- ▶ Web container and Object Request Broker (ORB) thread pools
- ▶ Session manager - HTTP Servlet sessions
- ▶ Transaction manager - Transactions
- ▶ JVM run time - Application server JVM
- ▶ JVMPi - JVM Profiler Interface data
- ▶ J2C - J2C connectors

To monitor the performance of a Web services application, you can use Tivoli Performance Viewer (TPV).

By viewing TPV data, administrators can determine which part of the application and configuration settings to change to improve performance, for example, you can view the servlet summary reports, enterprise beans, and Enterprise JavaBeans (EJB) methods to determine what part of the application to focus on. Then, you can sort these tables to determine which of these resources has the highest response time. Focus on improving the configuration for those application resources that take the longest response time.

Tivoli Performance View (TPV) is covered in more detail in 7.8.3, “Tivoli Performance Viewer” on page 133.

7.2.1 The performance review process

Solving a performance situation is frequently an iterative process because when one bottleneck is removed, the performance is now constrained by some other part of the system.

The process consists of three phases:

- ▶ Measuring system performance and collecting performance data
- ▶ Locating a bottleneck
- ▶ Eliminating a bottleneck

Measuring system performance and collecting performance data

Begin by choosing a benchmark that is based on a standard set of operations to run. This benchmark would be designed to exercise those application functions that experience performance problems. There are several sub topics to explore in your development of a performance benchmark:

Benchmark warm-ups	Complex systems frequently need a warm-up period to cache objects, optimize code paths, and so on. System performance during the warm-up period is usually much slower than after the warm-up period. The benchmark must be able to generate work that warms up the system prior to recording the measurements that are used for performance analysis. Depending on the system complexity, a warm-up period can range from a few thousand transactions to longer than 30 minutes.
Benchmarks under load	If the performance problem under investigation only occurs when a large number of clients use the system, then the benchmark must also simulate multiple users. Another key requirement is that the benchmark must be able to produce repeatable results. If the results vary more than a few percent from one run to another, consider the possibility that the initial state of the system might not be the same for each run, the measurements are made during the warm-up period, or that the system is running additional workloads.
Benchmark tools	Several tools facilitate benchmark development. The tools range from invoking an URL to script-based products that can interact with dynamic data that the application generates. IBM Rational has tools that can generate complex interactions with the system under test and simulate thousands of users. Producing a useful benchmark requires effort and needs to be part of the development process. Do not wait until an application goes into production to determine how to measure performance.
Benchmark output	The benchmark records throughput and response time results in a form to allow graphing and other analysis techniques. The performance data that is provided by WebSphere Application Server

Performance Monitoring Infrastructure helps to monitor and tune the application server performance. Request metrics is another source of performance data that WebSphere Application Server provides. Request metrics allow a request to be timed at WebSphere Application Server component boundaries, which enables a determination of the time that is spent in each major component.

Locating a bottleneck

Using the output from our benchmarks, we have information to analyze. For this discussion, we present three scenarios where we present suggested solutions to the issues at hand:

- ▶ Scenario 1: Poor performance occurs with only a single user

Suggested solution: Utilize metrics to determine how much each component is contributing to the overall response time. Focus on the component accounting for the most time. Use Tivoli Performance Viewer to check for resource consumption. More information on the Tivoli Performance Viewer is located in 7.8.3, “Tivoli Performance Viewer” on page 133.

- ▶ Scenario: Poor performance only occurs with multiple users

Suggested solution: Determine if any systems have high CPU, network, or disk utilization, and address those. For clustered configurations, check for uneven loading across cluster members.

- ▶ Scenario: None of the systems seem to have a CPU, memory, network, or disk constraints, but performance problems occur with multiple users

Suggested solutions: Check that work is reaching the system under test. Ensure that some external device does not limit the amount of work reaching the system. Tivoli Performance Viewer helps to determine the number of requests in the system.

A thread dump might reveal a bottleneck at a synchronized method or a large number of threads waiting for a resource.

Make sure that enough threads are available to process the work in the IBM HTTP, database and the application servers. Conversely, too many threads can increase resource contention and reduce throughput.

Garbage collection (GC) is a form of automatic memory management. The garbage collector attempts to reclaim garbage or memory used by objects that will never be accessed or mutated again by the application. Monitor for garbage collections with the Tivoli Performance Viewer.

Eliminating a bottleneck

Consider the following methods to eliminate a bottleneck:

- ▶ Reduce the demand

Reducing the demand for resources can be accomplished in several ways. Caching can greatly reduce the use of system resources by returning a previously cached response, thereby avoiding the work needed to construct the original response.

Application code profiling can lead to a reduction in the CPU demand by pointing out hot spots you can optimize.

- ▶ Increase resources

Modifying tuning parameters to increase some resources, can be utilized to eliminate bottlenecks. 7.4, “Performance tuning for the WebSphere Application Server” on page 124 provides examples of how WebSphere Application Server can be presented with selective increases in resources to improve over-all performance.

- ▶ Re-design your applications

Some critical sections of the application and server code might require synchronization to prevent multiple threads from running code simultaneously, which can lead to incorrect results. Synchronization preserves correctness, but it can also reduce throughput when several threads must wait for one thread to exit the critical code path. Synchronization delays can often be reduced by changing the code to only use synchronization when necessary or by reducing the path length of the synchronized code.

7.3 Performance tuning for TCP/IP

TCP/IP is the backbone for communication for SOA solutions. We now touch on a few important performance topics.

7.3.1 Important client-side SETSOCKOPT TCP/IP parameters

The important client-side SETSOCKOPT TCP/IP parameters are:

SO_KEEPALIVE The SO_KEEPALIVE parameter activates KEEPALIVE for this socket. The SO_KEEPALIVE option causes a packet (called a 'keepalive probe') to be sent to the remote system if a long time passes with no other data being sent or received. This packet is designed to provoke an ACK response from the peer, which enables detection of a peer that becomes unreachable (for example, powered off or disconnected from the net).

SO_LINGER The SO_LINGER parameter addresses the situation of a Socket Close that arrives before all data from a previous write was processed. We suggest setting SO_LINGER=Y and VALUE=10, which results in a return to the client code when an ACK is received from the host or after 10 seconds. Socket Close must not lose previously sent data.

TCP_NODELAY=DISABLE

The TCP_NODELAY=DISABLE parameter optimizes write performance. Buffer writes are allowed while waiting for an ACK from the previous write.

7.3.2 PROFILE.TCPIP specifications

In this section, we provide some z/OS environment specifications:

- ▶ Set the KEEPALIVEOPTIONS parameter, which is the time in minutes to send a packet so that a connection does not time out.
- ▶ Ensure that the specification of the port in the TCPIP profile data set has NODELAYACKS specified, as shown here:

```
PORT 8082 TCP NODELAYACKS
```

Changing this can improve throughput by as much as 50% (this is particularly useful when dealing with trivial workloads). This setting is important for good performance when running SSL.

- ▶ SOMAXCONN is the number of connection requests that can be queued because the IMS Connect has not yet issued the accept call.

7.3.3 BPXPRMxx member in SYS1.PARMLIB

There are a few parameters to examine:

- ▶ MAXSOCKETS sets a limit for the total number of sockets in a system
- ▶ MAXFILEPROC sets the maximum number of sockets for any job (for example, an IMS Connect)

7.3.4 XCF tuning

XCF tuning is a z/OS environmental item. MAXMSG sets the number of XCF signaling buffers. A XCF buffer shortage can be seen as an IMS Connect hang condition. This setting depends on the size and frequency of the messages and the performance of the signaling paths and systems that are involved in the message transfer.

7.3.5 The Domain Name System configuration

Using the *Domain Name System* (DNS) systems can look up host names, both within the private network and across the Internet. Ensure that the DNS configuration is optimized so that lookups for frequently-used servers and clients are being cached. Some of the parameters to be reviewed that relate to DNS are:

- ▶ Resolvertimeout: Resolvertimeout specifies the amount of time that the resolver waits for a response while trying to communicate with a name server. This value, which is configured in the TCPIP.DATA, has a default of 30 seconds.
- ▶ Time to Live: Time to Live (TTL) is a limit on the period of time, number of iterations, or transmissions in computer and computer network technology that a unit of data (for example, a packet) can experience before it is discarded. In IPv4, time to live is an 8-bit field in the Internet Protocol (IP) header.

Caching is sometimes related to the name server's TTL value. On the one hand, setting the TTL high ensures good cache hits. However, setting it high also means that if the daemon goes down, it takes a while for everyone in the network to be aware of it.

A good way to verify that your DNS configuration is optimized is to issue the **oping** and **onslookup** UNIX System Services commands. Make sure that they respond in a reasonable amount of time. Often, a mis-configured DNS or DNS server name causes delays of 10 seconds or more.

7.3.6 Send and receive buffer specifications

The TCP/IP socket connection send and receive buffer sizes define the receive window. The receive window specifies the amount of data that can be sent and not received before the send is interrupted. If too much data is sent, it overruns the buffer and interrupts the transfer. If the receive window size for TCP/IP buffers is too small, the receive window buffer is frequently overrun and the flow control mechanism stops the data transfer until the receive buffer is empty.

Flow control can consume a significant amount of CPU time and result in additional network latency as a result of data transfer interruptions. We recommend that you increase buffer sizes to avoid flow control under normal operating conditions. A larger buffer size reduces the potential for flow control to occur and results in improved CPU utilization. However, a large buffer size can have a negative effect on performance, in some cases. If the TCP/IP buffers are too large and applications are not processing data fast enough, paging can increase. The

goal is to specify a value that is large enough to avoid flow control, but not so large that the buffer accumulates more data than the system can process.

Increase the size of the TCPIP send and receive buffers from the default of 16 K to at least 64 K, which is the size of the buffers including control information beyond what is present in the data that you are sending in your application. Tcpsendbfrsize and Tcprcvbfrsize are the parameters that are used for these settings.

The finwait2 time

In the most demanding environments, you might find that even defining 65 K sockets and file descriptors does not give you enough *free* sockets to run at 100% throughput. When a socket is closed abnormally (for example, no longer needed) it is not made available immediately; instead, it is placed into a state called finwait2 (this is what shows up in the `netstat -s` command). It waits there for a period of time before it is made available in the free pool. The default for this is 600 seconds, which you must reduce for improved performance.

7.3.7 Hipersockets

Mainframe HiperSockets™ is a technology that provides high-speed TCP/IP connectivity within a central processor complex. It eliminates the need for any physical cabling or external networking connection between servers that are running in different logical partitions (LPARs).

The communication is through the system memory of the processor, so servers are connected to form an *internal LAN*.

Figure 7-1 shows the position of the HiperSockets support. Notice that z/OS is not the only operating system that is running on a mainframe host that can take advantage of HiperSockets. The other operating systems are z/VM® and Linux.

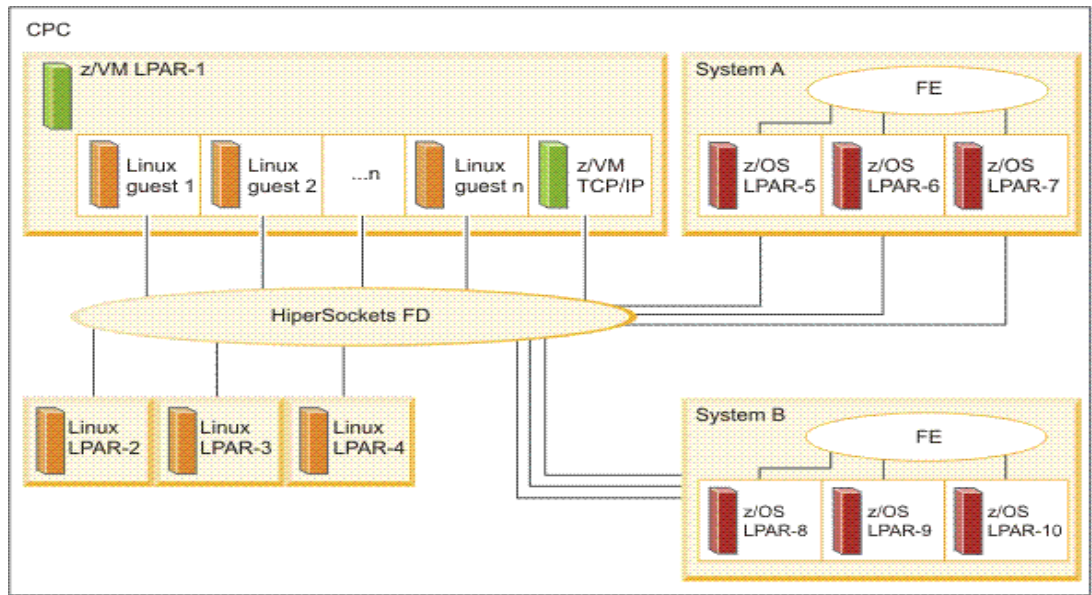


Figure 7-1 HiperSockets positioning

7.4 Performance tuning for the WebSphere Application Server

WebSphere Application Server has many performance tuning options, and we touch on a few. For more information, refer to *Monitoring WebSphere V5.1 Application Performance on z/OS*, SG24-6825.

7.4.1 Queuing network tuning

There are many interconnected queues in a SOA environment. There are network, Web server, application server, and data source queues, amongst others. It is important to minimize the number of requests in the application server (WebSphere Application Server) queue by configuring the upstream (front end) queues slightly larger than the downstream (back end) queues because as long as at each point upstream there is some work waiting to be executed, each component downstream is fully utilized.

To achieve the tuning of the queuing network, the work of a Web server needs to be limited. One method of limiting the work of a Web server is by controlling the MaxActiveThreads setting. Use this caching proxy directive to set the maximum number of threads that are active at one time. If the maximum is reached, the server holds new requests until another request finishes and threads become available. Generally, the more power a machine has, the higher the value that is set for this directive. If a machine starts to spend too much time on overhead tasks, such as swapping memory, try reducing this value:

- ▶ If the server is running in scalable server mode, the value 100 or less is recommended.
- ▶ In standalone mode, a value of 150 or less is recommended.
- ▶ The value must be lower than the z/OS UNIX System Services BPXPRMxx member setting for MAXTHREADTASKS.

7.4.2 WebSphere plug-in performance

The main role of the WebSphere plug-in is to direct traffic from the HTTP server to the application server instances and maintain session affinity. There are a number of settings that you can configure in the plug-in to improve performance. You can manipulate those settings using the WebSphere Administrative Console.

Refreshinterval

Refreshinterval is the time interval (in seconds) at which the plug-in checks the configuration file for updates/changes. The default value of 60 seconds is ideal for a development environment, and 0.6 seconds is a good starting point for production environments.

LogLevel

LogLevel sets the details of the log messages that the plug-in writes. There are four values: Error, Trace, States, and Warn. Error is the default and the recommended value. Do not use the Trace option in a normally-functioning environment.

Retryinterval

Retryinterval is the length of time (in seconds) that the plug-in waits before trying to connect to a server that was marked temporarily unavailable. The default value is 60 seconds. Lower this value to increase throughput under heavy load conditions. Note that lowering the value too much can result in increasing latency and decreased overall throughput.

ConnectTimeout

ConnectTimeout specifies the number of seconds that you want the plug-in to wait for successful connection with the application server; otherwise, it is marked as unavailable. A value greater than 0 specifies the number of seconds that you want the plug-in to wait for successful connection. The suggested value is 15.

LoadBalance

LoadBalance specifies the load balancing option that the plug-in uses while sending requests to the application servers that are associated with the HTTP Server. The round robin option is the recommended option.

LoadBalanceWeight

LoadBalanceWeight is the weight that is associated with the server when the plug-in does weighed round-robin load balancing. We recommend that when a server is shut down for a period of time, to set this value to 0. The plug-in can then reset the weights of the servers that are still running and maintain proper load balancing.

Administrative console settings

There are a number of settings that you can configure to improve performance, which you can manipulate using the WebSphere Administrative Console. Many of the settings depend on what the applications are doing and the amount and type of workload.

Tracing

To view or set your trace settings, use the WebSphere Administrative Console:

1. Click **Environment** → **Manage WebSphere Variables**.
2. On the Configuration Tab, check for any of these variables in the name field, and observe the variable setting in the value field.
3. Unless a particular problem demands it, disable or minimize any tracing to improve performance. Do not enable Application and JRAS tracing unless requested during problem analysis.

Workload profile

The WORKLOAD_PROFILE parameter establishes the number of threads in a single servant region. You can find The WORKLOAD_PROFILE parameter in the administrative console by selecting:

Servers → **Application Servers** → **server name** → **ORB service** → **Advanced Settings**

The values for the WORKLOAD_PROFILE parameter are:

- ▶ IOBOUND: The number of threads is three times the number of CPUs available, with a minimum of five and a maximum of 30.
- ▶ CPUBOUND: Provides for a number of threads that are equal to the number of CPUs available, minus one, with a minimum of three. This setting is for applications that are CPU intensive.
- ▶ LONGWAIT: Provides a fixed number of 40 threads. This setting is appropriate for applications where the majority of processing occurs in a back-end subsystem, such as an IMS transaction.
- ▶ ISOLATE: Provides only one thread. This setting is only for applications that must run single threaded. Significant scalability problems can arise from using this value; therefore, take care when using it.

Security

Security is important but it does add additional overhead for each transaction:

- ▶ Use the minimum number of EJBROLES on methods. Java EE Security provides a mechanism called *EJBRoles* that you can use to provide security for applications that are running in Java EE-compliant application servers, which includes WebSphere Application Server.
- ▶ If you do not need it, disable Java 2 security.
- ▶ Use the relevant level-of-security authentication according to your security needs.

LogStream compression

If your application components are using recovery log services, the LogStream variable sets how frequently the recovery log service attempts to compress the logstreams:

- ▶ If you are not using the services, consider disabling the compression (value = 0).
- ▶ If using compression, set the value high.

Object request broker

The Object request broker (ORB) describes the way parameters are passed between EJB clients and the EJB when they sit in the same classloader. Set the value to “pass by reference”.

Monitoring policy ping interval

Monitoring policy ping interval is how often the Node Agent sends the ‘isAlive’ SOAP messages:

- ▶ Decreasing the interval value detects failures sooner.
- ▶ Increasing the value reduces the frequency of pings, which reduces system overhead.

File synchronization

File synchronization is the interval (in minutes) between automatic synchronization of changes across multiple nodes:

- ▶ In an environment with infrequent changes, consider a high number.
- ▶ In a production environment, consider disabling automatic synchronization.

Pre-compiled JavaServer Pages

Using pre-compile JavaServer Pages (JSPs) shortens the time of the first invocation of JSPs and reduces compilation-related disk operations.

You can pre-compile JSPs at different points of the deployment process:

- ▶ In Rational Application Developer or other development tooling
- ▶ In the administration console when deploying an application
- ▶ Run a JSP Batch compiler tool, after the application was deployed

URL invocation cache

The URL invocation cache holds information for mapping request URLs to servlet resources. A larger cache uses more of the Java heap, so you might also need to increase the maximum Java heap size.

7.4.3 Using the dynamic cache service to improve performance

Caching the output of servlets, commands, and JavaServer Pages (JSP) improves application performance. WebSphere Application Server consolidates several caching activities, including servlets, Web services, and WebSphere commands, into one service called the dynamic cache. These caching activities work together to improve application performance and share many configuration parameters that are set in the dynamic cache service of an application server:

- ▶ Enable the dynamic cache service globally. To use the features that are associated with dynamic caching, you must enable the service in the administrative console:
 - WebSphere Application Server inspects the HTTP request to determine whether or not an incoming message can be cached based on the cache policies that are defined for an application.
- ▶ Monitor the results of your configuration using the dynamic cache monitor:
 - The dynamic cache monitor is a Web application that displays simple cache statistics, cache entries, and cache policy information for servlet cache instances.

7.4.4 Workload Manager tuning for WebSphere Application Server

WebSphere Application Server uses the z/OS WLM to manage workload distribution between servers and to increase or decrease the number of servants in response to workload activity. Tuning WLM can improve WebSphere performance significantly. Some of the important Workload Manager configuration settings are:

- ▶ Minimum number of instances: This is the number of servants that are always up and running, regardless of what the workload is. Make the value higher than 1.
- ▶ Maximum number of instances: Setting this variable to 0 allows WLM to start up as many servants as needed.
- ▶ Classify controllers and daemons as SYSSTC or lower.
- ▶ Classify servants high, but lower than controllers.
- ▶ The work can be qualified by those parameters:
 - CN: Collection Name (server_generic_short_name for WAS on z/OS)
 - SI: Server Instance name (server_specific_short_name)
 - TC: Transaction Class mapping file for the server (helps us differentiate performance objects between multiple transactions running on the same WebSphere server)
 - User ID: User Name (the user name under which the work request is running)
- ▶ When defining TC, also configure:
 - Wlm_minimumSRCount = Number of expected service classes
 - Wlm_maximumSRCount = Wlm_minimumSRCount * Number of expected service classes, or, at least Wlm_minimumSRCount
- ▶ Velocity goals for application work are not meaningful, so avoid them.

7.4.5 Performance considerations for WebSphere connectors

In WebSphere on z/OS, connections are used to connect to Enterprise Information Systems, such as IMS. To reduce the strain on application resources, using WebSphere Application Server administrators can establish a pool of back-end connections that are used by all user

requests. By configuring connection pooling, the response time and performance of any application that uses connections can be improved.

Pool connection properties

The important properties that are associated with back-end connection pools are:

- ▶ **Connection Timeout:** The Connection Timeout property specifies the time (in seconds) a request for connection can wait (when there are no free connections on the pool and new ones cannot be created). When the waiting time exceeds the connection time-out value, the pool manager initiates a ConnectionWaitTimeout exception. If connection time-out is set to zero, the pool manager waits as long as necessary until a connection becomes available.
- ▶ **Maximum Connections:** The Maximum Connections property specifies the maximum number of physical connections that you can create in a pool. If Maximum Connections is set to 0, the connection pool is allowed to grow infinitely, and the Connection Timeout is ignored.
- ▶ **Minimum Connections:** The Minimum Connections property specifies the minimum number of physical connections you can create in a pool.
- ▶ **Unused Timeout:** The Unused Timeout property specifies the interval (in seconds) after which an unused or idle connection is discarded. Unused physical connections are only discarded if the number of the connections exceed the Minimum Connections value.
- ▶ **Aged Timeout:** The Aged Timeout property specifies the interval (in seconds) after which a physical connection is discarded:
 - When set to 0, a physical connection remains in the pool indefinitely.
 - For optimal performance, make the value higher than the value of Reap Time.
 - If the connection is involved in a transaction and the Aged Timeout is reached, the connection is closed immediately after the transaction completes.
- ▶ **Purge Policy:** The Purge Policy property specifies how to purge a connection when a stale connection or fatal error is detected.
- ▶ **Entire Pool:** All connections in the pool are marked stale. Connections that are not used are closed immediately.
- ▶ **FailingConnectionOnly:** Only the connection that causes the stale connection exception is closed. If a valid connection is closed also, the recovery process from the application perspective is more complicated.
- ▶ **Reap Time:** Reap Time property specifies the interval (in seconds) between runs of the pool maintenance threads. Make the value less than the value of Unused Timeout and Aged Timeout (when it runs it discards any connections that remain unused for longer than the Unused Timeout value until it reaches the value of Minimum Connection). The smaller the value is, the greater the accuracy of Unused Timeout and Aged Timeout.

7.5 IMS Connect performance parameters

Within the HWSCFG IMS Connect configuration member there are parameters of performance interest:

- ▶ Set ECB=Y to posts an ECB when there is work to do.
- ▶ MAXSOC = xxxx sets the maximum number of concurrent sessions for all of IMS Connect. Specify a large enough value to support concurrent throughput requirements.

- ▶ Set IPV6=Y for better performance, even if the network itself is not at IPV6 level.
- ▶ The NODELAY parameter is used for output packets from the IMS Connect side. Specify Y to enhance IMS Connect performance and increase throughput because it forces a socket to send the data in its buffer without having to wait for an ACK from the client's TCP/IP.

7.6 IMS Connect Extensions

IMS Connect is the premier pathway for accessing IMS applications across TCP/IP. IMS Connect Extensions for z/OS enhances this function, reducing operational costs and providing users with a quicker return on investment.

The product simplifies the development, debugging, and tuning of TCP/IP clients, helps you evaluate and meet your service level agreements (SLAs), and allows you to control all of your IMS Connect instances from a single point. You can also use it to assist the support staff to better manage performance, accurately plan capacity, and dynamically manage workloads. IMS Connect Extensions collect internal event data from IMS Connect, which you can use with the IMS Performance Analyzer for z/OS. Together, these products produce detailed performance reports with useful information that includes:

- ▶ Internal and external transit times and latencies for IMS Connect transactions.
- ▶ Message activities, for example, OTMA, READ and XMIT exits, read and write socket utilization, and security and commit confirm elapsed times.
- ▶ Network activity, which includes port and socket utilization, the amount of data that is processed and accepted, and read and write socket request counts.

Also IMS Connect Extensions help the support staff to manage IMS Connect workloads by providing flexible control of input message traffic. It can balance workloads between eligible data stores and reroute all messages to an alternate data store when a given data store is unavailable.

For more information about IMS Connect Extensions, refer to Chapter 9, “IMS Connect and IMS Connect Extensions” on page 167.

7.7 Language Environment performance discussion

Language Environment (LE) provides a common runtime environment for high-level programming, such as C, C++, or Cobol.

Some of the configuration hints and tips concerning LE are:

- ▶ Load LE and C++ reentrant runtimes into LPA:
 - SYS1.PARMLIB(LPALSTxx) must have the following defined:
 - CEE.SCEELPA
 - CBS.SCLBDLL

- ▶ Non-reentrant LE routines must be in the linklist:
 - SYS1.PARMLIB(LNKLSTxx) must include CEE.SCEERUN.
- ▶ Ensure that LE data sets CSEERUN and SCEERUN2 are authorized to enable XPLINK:
 - Processes that run the client ORB (they start the JVM), must run with XPLINK(ON).
 - For best performance, compile applications that use JNI™ services with XPLINK enabled.

7.8 Performance monitoring tools

There are many tools that you can use to monitor for changes in your environment and reduce opportunities for degraded performance.

Table 7-1 presents a summary of the tools that we cover in this section.

Table 7-1 Summary of performance monitoring tools

Tool	Functionality
Resource Measurement Facility	Collecting performance data for z/OS and sysplex environments to monitor the systems performance behavior
Eclipse Test and Performance Tools Platform (TPTP)	Building test and performance tools and addressing the entire test and performance life cycle
Tivoli Performance Viewer	Monitoring the WebSphere Application Server and providing graphics and charts on various performance data through the administrative console
Tivoli Composite Application Monitor	Monitoring WebSphere Application Server and Java EE applications, middleware adapters and transports, database calls, and back-end systems, such as IMS and CICS
WebSphere Business Monitor	Offering business activity monitoring through portals that enable you to monitor different aspects of business performance.
Tivoli Omegamon XE for IMS Connect	Analyzing z/OS resources used by IMS address spaces and internal IMS resource usage
Rational Performance Tester for z/OS	Pinpointing system bottlenecks before application deployment by simplifying the creation, execution, and results analysis of multi-user performance tests through test recording and test scheduling
JMeter	Load testing tool for analyzing and measuring the performance of a variety of services, with a focus on Web-based applications
IMS Performance Analyzer	Providing offline performance analysis across all IMS subsystems
IMS Problem Investigator	Formatting and analyzing IMS log data and connecting this data to SMF, DB2, and MQ logs

7.8.1 Resource Measurement Facility

The Resource Measurement Facility (RMF) analyzes the z/OS environment using data that is written to the System Management Facility (SMF) data sets.

RMF is designed to ease the management of single or multiple system workloads and to enable faster reaction to system delays. System programmers are supported by several new reports that ease their work and helps them to optimally tune their system, which consequently, leads to fewer workload problems, and most importantly, increases system and operator productivity:

- ▶ You can use RMF reports based on (near) real-time monitoring or historical data
- ▶ RMF provides:
 - An outside view of the operating system resources usage
 - A closer look on WLM information (meeting the goals, and so on)
- ▶ The RMF components are:
 - RMF monitor I, II and III running on z/OS
 - RMF sysplex data server and APIs running on z/OS
 - RMF PM running on Windows and Linux
 - RMF Spreadsheet reporter operating on Windows

RMF Performance Monitor Java Technology Edition

Using the RMF Performance Monitoring — Java Technology Edition (RMF PM) you can monitor the performance of the z/OS environment from one focal point (a workstation that is running Windows 2000, Windows XP, Windows ME, or Linux) through a TCP/IP interface to one or more z/OS sysplexes.

RMF PM takes its input data from a single data server on one system in the sysplex, which gathers the data from the RMF Monitor III on each MVS image. You can analyze and monitor data from the present or the recent past.

Figure 7-2 on page 132 shows an example of the workstation client set of windows.

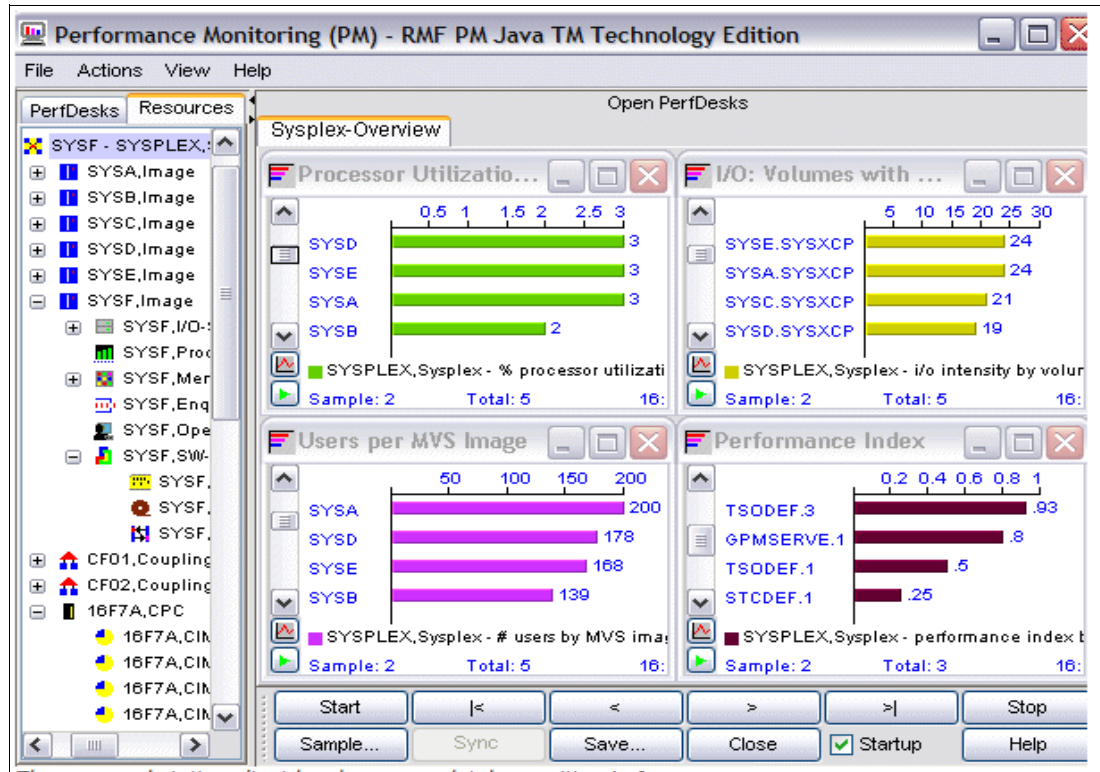


Figure 7-2 RMF PM workstation client set of windows

The information that RMF PM provides is:

- ▶ General performance data
- ▶ Performance data for jobs
- ▶ Performance data for systems that are running in goal mode
- ▶ WLM performance data, such as WLM workloads, WLM Service Classes and periods, and WLM report Classes

Using RMF PM, you can create unique views that display both historic and real-time information.

7.8.2 Eclipse Test and Performance Tools Platform

The Eclipse Test and Performance Tools Platform (TPTP) provides an open platform, supplying frameworks and services that allow software developers to build test and performance tools that can be easily integrated with the platform and with other tools. TPTP is supplied from the Eclipse Foundation.

TPTP addresses the entire test and performance life cycle, from early testing to production application monitoring, which includes test editing and execution, monitoring, tracing and profiling, and log analysis capabilities.

Why use TPTP

Writing the Java code for an application or servlet is just the first stage in the much longer process that is required to deliver robust production-quality programs. Code must often be profiled to remove bottlenecks that impede performance and to remove wasteful or inadvertent use of resources, especially memory. Code must also be monitored to pinpoint

failures, but also to identify usage patterns, opportunities for further enhancement and optimization, and attempted and actual intrusions.

For more information, refer to the Web site:

<http://www.eclipse.org/tptp/index.php>

7.8.3 Tivoli Performance Viewer

The Tivoli Performance Viewer (TPV) was previously named the Resource Analyzer. TPV monitors the WebSphere Application Server and provides graphics and charts about various performance data from within the administrative console:

- ▶ It is based on a Java application and is part of the WebSphere Application Server administrative console.
- ▶ TPV can run in two topologies, a single server in the JVM or in a network deployment environment.

Logging performance data with TPV

TPV provides an easy way to store real-time data for system resources, WebSphere Application Server pools and queues, and applications. The data is stored in log files for later retrieval. Users can start and stop logging while viewing current activity for a server and later replay this data.

Monitoring current server activity

You can view real-time data on the current performance activity of a server using TPV in the administrative console:

- ▶ Use the performance advisors to examine various data while your application is running. The performance advisor in TPV provides advice to help tune systems for optimal performance by using collected PMI data.
- ▶ Configure user and logging settings for TPV. These settings can affect the performance of your application server.
- ▶ View summary reports on servlets, Enterprise JavaBeans (EJB) methods, connections, and thread pools in WebSphere Application Server.
- ▶ View performance modules that provide graphs on system resources, such as CPU utilization, WebSphere pools and queues, such as database connection pools, and on customer application data, such as servlet response time. In addition to providing a viewer for performance data, using TPV you can view data for other products or customer applications that implemented custom PMI.

7.8.4 Tivoli Composite Application Monitor

The Tivoli Composite Application Monitor (ITCAM) provides monitoring for WebSphere Application Server and Java EE applications, middleware adapters and transports, database calls, and back-end systems, such as CICS and IMS.

With ITCAM, you can stay ahead of composite application and Web Services performance, availability and client-side response times by finding and fixing IT application-resource problems before your users feel an impact:

- ▶ IBM Tivoli Composite Application Monitor for WebSphere (ITCAM for WebSphere) evolved from several other products, such as WSAM and IBM Tivoli Omegamon XE for WebSphere

- ▶ ITCAM is a distributed application and consists of a managing server and one or more data collectors. ITCAM collects its data from JVMTI, PMI in WebSphere, and SMF in z/OS
- ▶ ITCAM reports on:
 - CPU utilization of application servers
 - Memory and EJB usage
 - WebSphere pools and queues

ITCAM contains a managing server that controls and coordinates the data collectors and the data collectors that run inside the application servers.

7.8.5 WebSphere Business Monitor

WebSphere Business Monitor is a Web application that is deployed and runs under WebSphere Process Server 6.0. It displays dashboards, which are containers (portals) that enable you to monitor different aspects of business performance.

Dashboards serve a wide audience, which includes all line-of-business and systems management users, and business executives, which enables them to:

- ▶ Monitor and manage business performance indicators.
- ▶ Personalize the analysis and display of business performance reports and compress information to focus on the business objectives and the *key performance indicators* (KPIs).
- ▶ View business-critical information graphically using visual cues, such as color, to improve the probability of timely problem determination and the speed of decision making.
- ▶ Visualize performance data, such as KPIs and metrics, which can be summarized in reports and graphs.
- ▶ Analyze and investigate business situations by using drill-down capabilities to trace situations to individual events and inspect event details.
- ▶ Set up actions and alerts that are part of the management phase of a business performance management solution.

Figure 7-3 on page 135 shows an example of a portal-based, fully-customized dashboard, which includes items, such as alerts, corporate objectives, and so on.

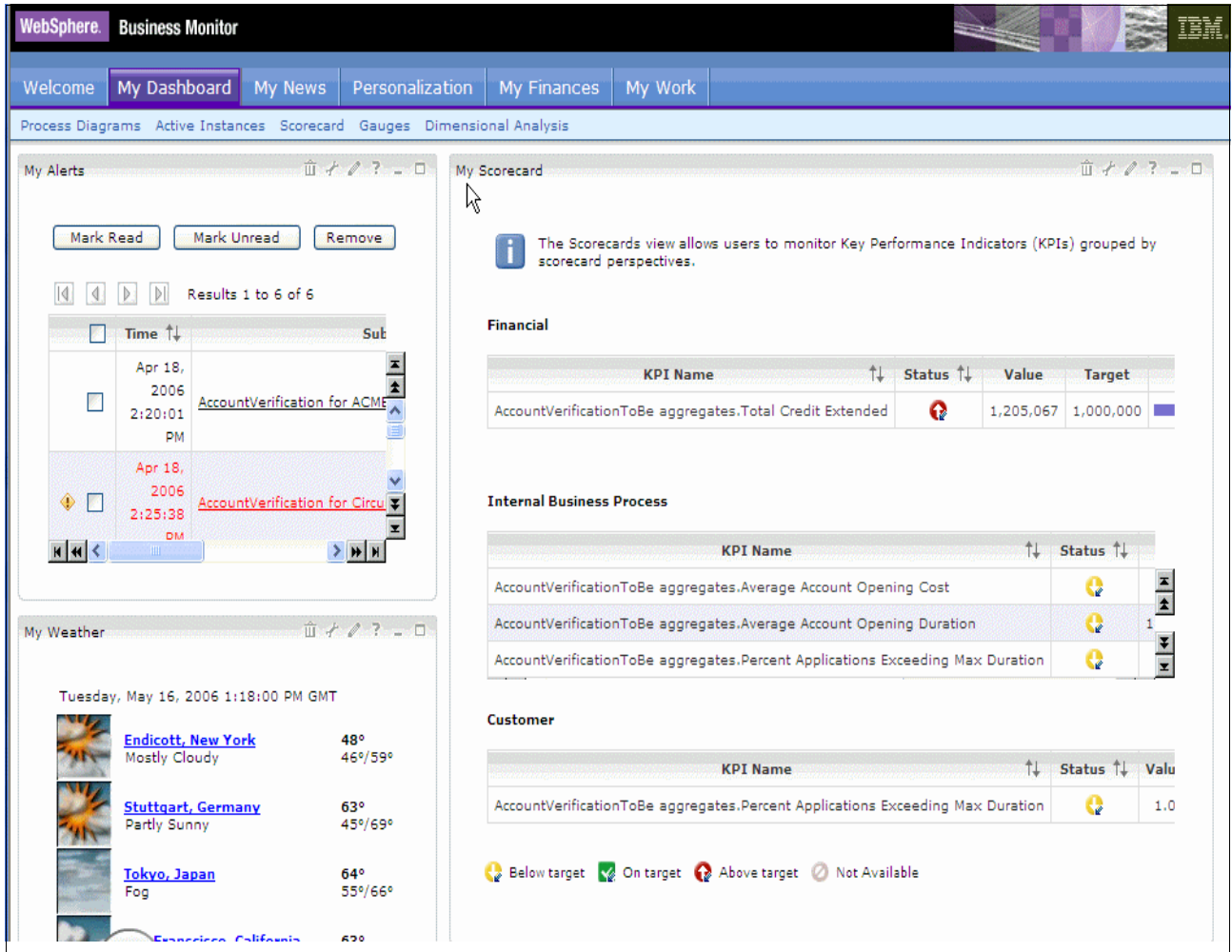


Figure 7-3 A fully customized WebSphere Business Monitor dashboard

Figure 7-4 on page 136 displays the alerts that were set up to send notifications to responsible staff.

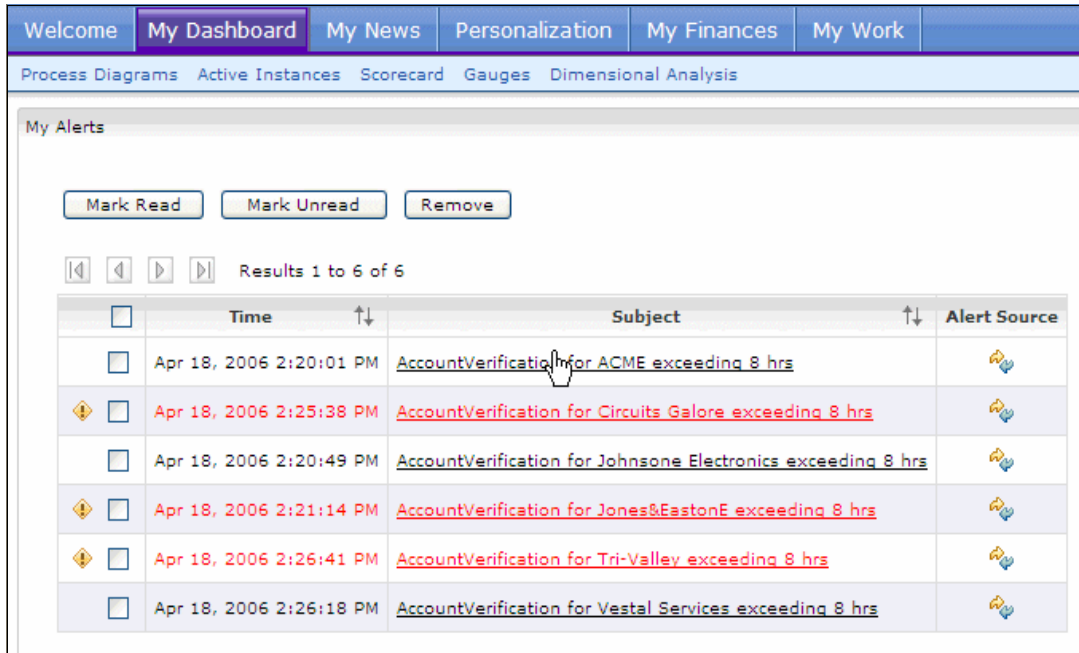


Figure 7-4 Sample of alerts from the WebSphere Business Monitor dashboard

Figure 7-5 displays the analysis of trends and aggregated data across the lifetime of the process. It also shows the KPIs compared to the expected business objectives.

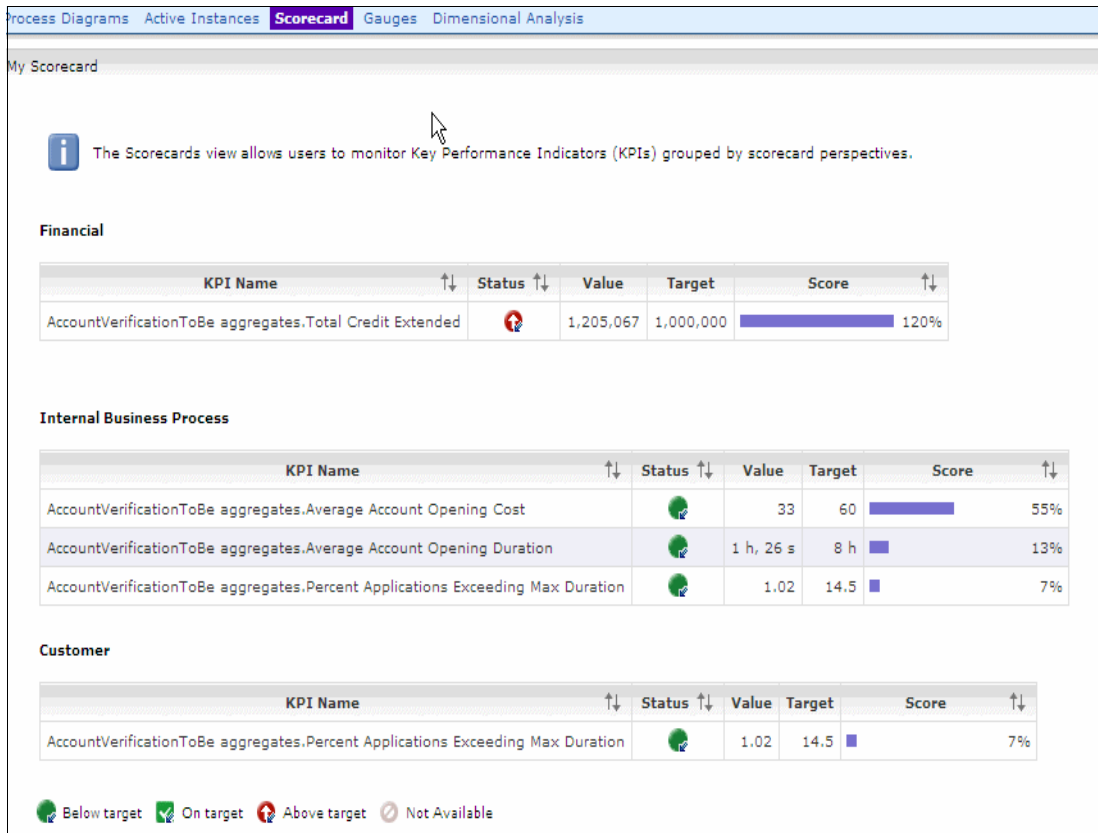


Figure 7-5 Trends across the lifetime of a process

Figure 7-6 displays the average account opening cost compared to the expected range.

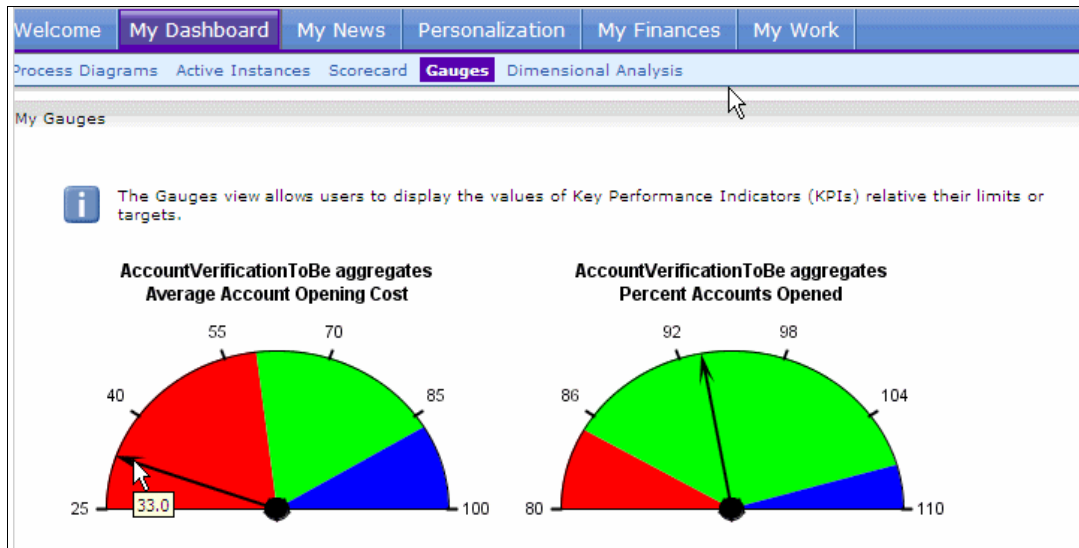


Figure 7-6 WebSphere Business Monitor dashboard gauges

Figure 7-7 displays WebSphere Business Monitor's ability to allow for drilling up and down into the data based on a time frame. Abode PDF files can be generated to include data for reporting purposes.

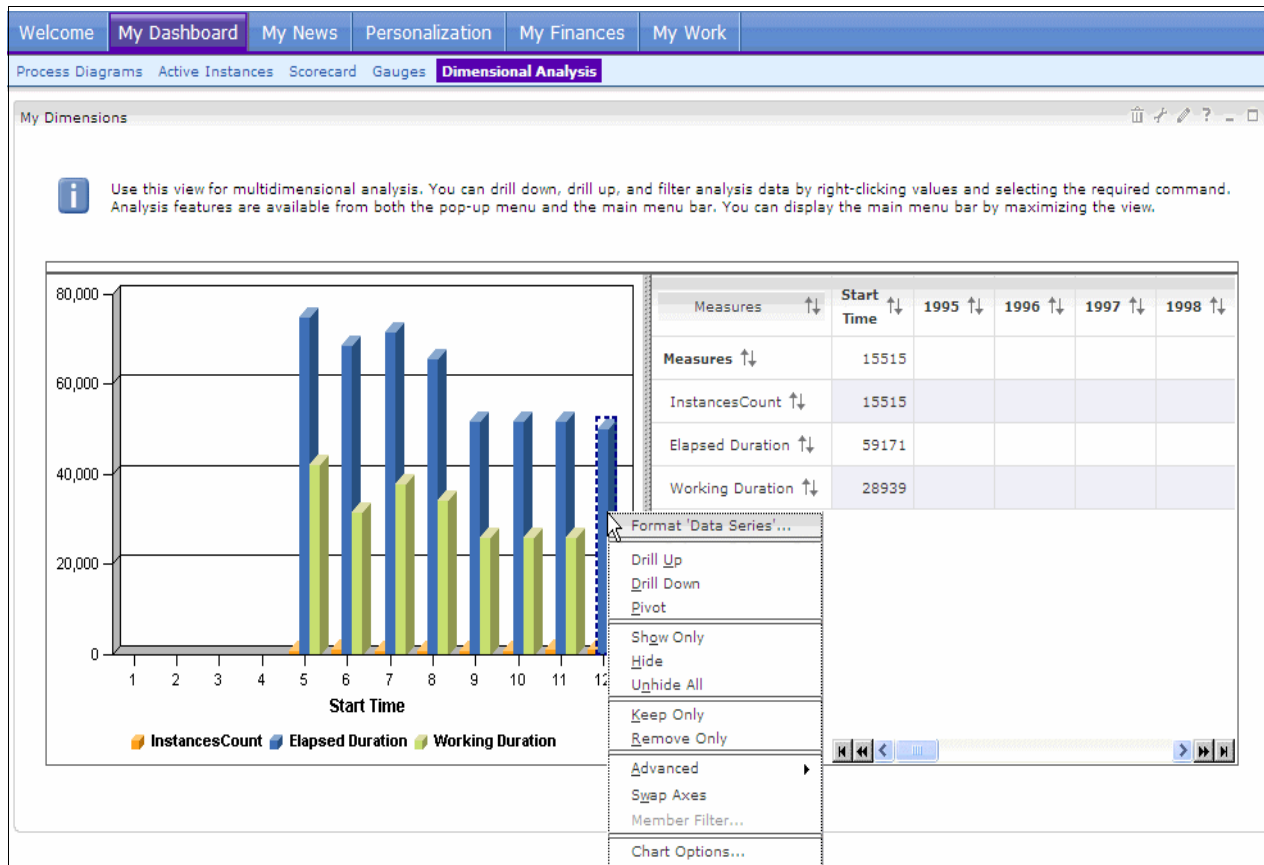


Figure 7-7 Drilling into the data based on a time frame

7.8.6 Tivoli Omegamon XE for IMS Connect

The Tivoli Omegamon XE tool analyzes z/OS resources that are used by IMS address spaces and internal IMS resource usage, such as IMS and database buffer pools. Within IMSplex environments, using Tivoli OMEGAMON XE for IMS you can see coupling facility structure statistics, shared queue counts, and database lock conflicts that help you stay ahead of potential delays or outages.

Viewing response time details by component

Extensive metrics let you see performance information on message flow through the IMS Connect function. You can see response time details by component, which includes:

- ▶ Input pre-OTMA
- ▶ Input read socket
- ▶ Input read exit
- ▶ Input system authorization facility (SAF)
- ▶ Process OTMA

The product summarizes response times by transaction, data store, TCP/IP port, and client and user IDs. Metrics let you see average and maximum response times, message counts, component average response times, messages processed per second, acknowledgement code (ACK) or negative acknowledgement (NAK), and Resume TPIPEs. TCP/IP usage statistics help to tune the z/OS communications environment.

To manage your IMS workloads effectively, Tivoli OMEGAMON XE for IMS includes more than 25 IMS Connect exception events that are formatted to help you more easily find and analyze problems.

Software requirements

The software requirements for Tivoli OMEGAMON XE for IMS are:

- ▶ IMS, Versions 9.1 and above
- ▶ IMS Connect Extensions, Version 1.2 and above
- ▶ z/OS, Version 1.4 and above

7.8.7 Rational Performance Tester for z/OS

Rational Performance Tester (RPT) for z/OS helps teams to pinpoint system bottlenecks before application deployment by simplifying the creation, execution, and results analysis of multi-user performance tests. Tests are recordings of a user's activity within a Web browser, and no programming knowledge is required to understand and modify these tests. Using an intuitive graphical test scheduler, you can then organize your tests to accurately simulate the different types of users and user activities that the application under test supports.

During test execution, although emulating the desired number of concurrent users, Rational Performance Tester for z/OS generates reports that clearly highlight poorly performing Web pages, URLs, and transactions; therefore, you can expose performance problems in even the most complex systems, which increases the opportunity for problem capture and repair before the system goes live.

7.8.8 JMeter

JMeter is an Apache Jakarta project that you can use as a load testing tool for analyzing and measuring the performance of a variety of services, with a focus on Web-based applications.

JMeter can generate SOAP and XML-RPC type requests, effectively functioning as a Web services client to stress test any server-side deployment.

You can use JMeter as a unit test for JDBC database connections, FTP, LDAP, Web services, JMS, HTTP, and generic TCP connections.

Use JMeter to test performance, both on static and dynamic resources (files, Servlets, Perl scripts, Java Objects, Data Bases and Queries, FTP Servers, and more). It can be used to simulate a heavy load on a server, network, or object to test its strength or to analyze overall performance under different load types.

Figure 7-8 displays the JMeter 2.1 support of HTTP, HTTPS, FTP, SOAP, JMS, JNDI, JDBC, and LDAP.

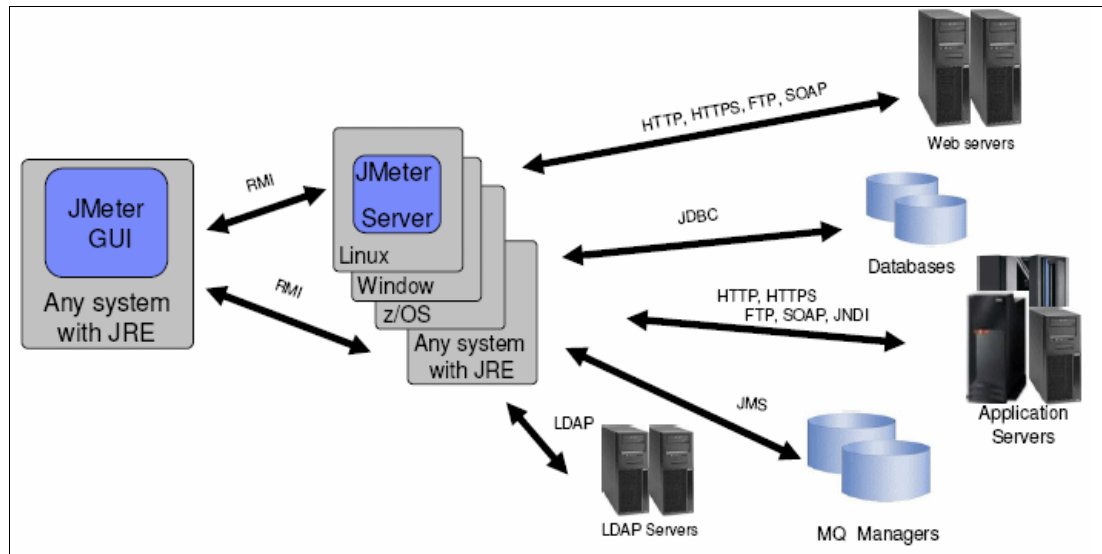


Figure 7-8 Positioning of Jmeter

7.8.9 IMS Performance Analyzer

The value that the IMS Performance Analyzer for z/OS Version 4.1 provides is end-to-end IMS Connect and IMS log reporting, revealing a transaction's life cycle through IMS Connect, and IMS. IMS log, monitor, IMS Connect event, and OMEGAMON TRF data can be processed to provide comprehensive reports.

Figure 7-9 on page 140 illustrates the wide range of input sources that can supply IMS PA. You can utilize both an ISPF dialog and batch commands to best manage your reporting requirements.

IMS PA provides comprehensive reporting from the IMS Connect performance and accounting data that IMS Connect Extensions for z/OS (5655-K48) collects. Summary and detailed reports analyze IMS Connect transaction internal and external transit times and latencies, highlighting critical events for message processing.

For more information, refer to the Web site:

<http://www-01.ibm.com/software/data/db2imstools/imstools/imspa.html>

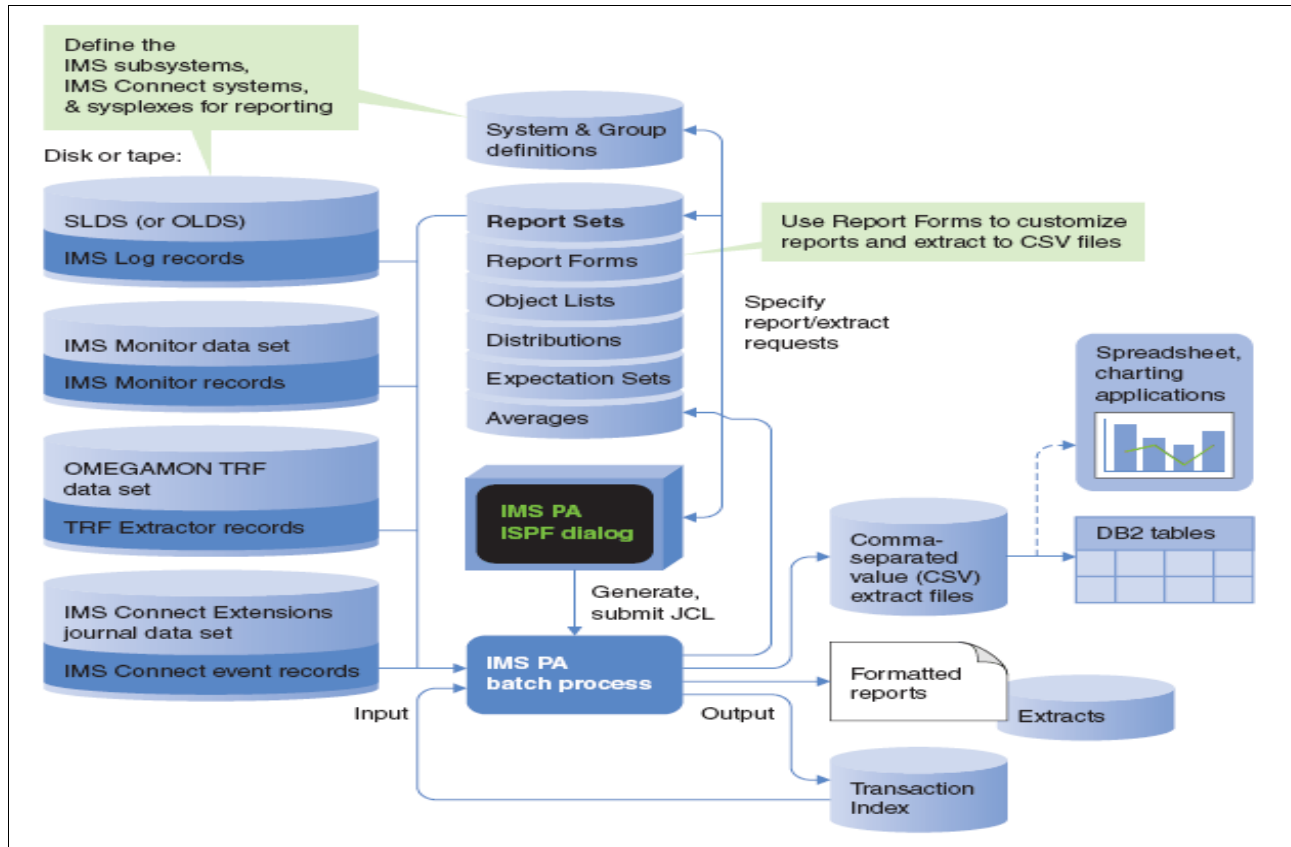


Figure 7-9 An overview of IMS Performance Analyzer operation

7.8.10 IMS Problem Investigator

Using the IMS Problem Investigator for z/OS Version 2.1 you can merge several log files into a single logical view for a complete end-to-end picture of a transaction's life cycle.

The types of records that the IMS Problem Investigator supports are:

- ▶ IMS log
- ▶ IMS Transaction Index created by IMS Performance Analyzer
- ▶ IMS monitor and DB monitor files
- ▶ Common Queue Server (CQS) log stream
- ▶ IMS Connect event data collected by IMS Connect Extensions
- ▶ OMEGAMON Transaction Reporting Facility (TRF) log and extracts
- ▶ DB2 log
- ▶ WebSphere MQ log extracts
- ▶ SMF – IRLM Long Lock records

You can analyze these records through an ISPF dialog, batch reports, and REXX programming services, and you can create filtered extracts for more efficient problem investigation. By tracking and displaying only those records that are associated with a selected transaction, you can develop a clear picture of that transaction's flow.

Figure 7-10 on page 141 illustrates a high-level overview of the input sources, processing, and outputs that are associated with the IMS Problem Investigator.

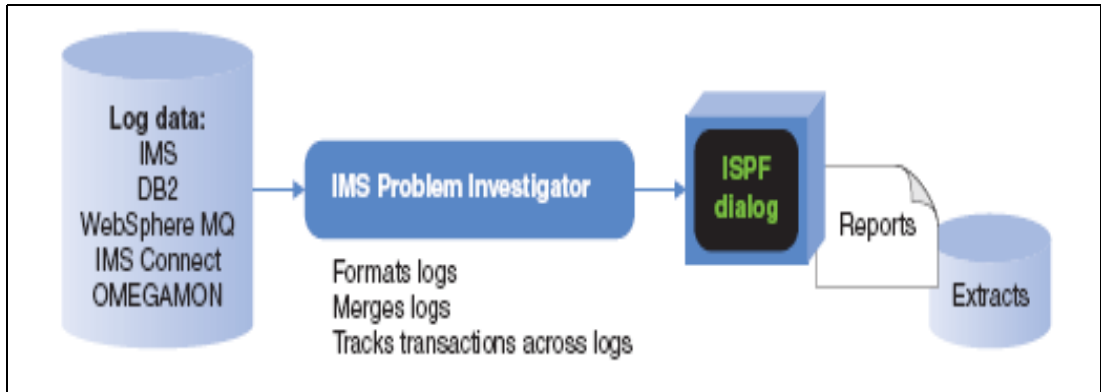


Figure 7-10 IMS Problem Investigator input, activity and output

For more information, refer to the Web site:

<http://www-01.ibm.com/software/data/db2imstools/imstools/imsprobleinvest.html>

This concludes our discussion about performance tooling for IMS SOA environments.



Part 3

SOA enhancements in IMS and the IMS SOA Integration Suite

The available IBM Redbooks that describe the earlier enhancements that improved the IMS SOA solution set are:

- ▶ *IMS Connectivity in an On Demand Environment: A Practical Guide to IMS Connectivity*, SG24-6794
- ▶ *IBM IMS Version 10 Implementation Guide A Technical Overview*, SG24-7526.

In part 3, we introduce enhancements that were made available after the publishing of that material.

The chapters in this part are:

- ▶ Chapter 8, “IMS SOA enhancements” on page 145
- ▶ Chapter 9, “IMS Connect and IMS Connect Extensions” on page 167
- ▶ Chapter 10, “The IMS SOAP Gateway” on page 203
- ▶ Chapter 11, “The IMS TM Resource Adapter” on page 225
- ▶ Chapter 12, “MFS Web solutions” on page 245
- ▶ Chapter 13, “IMS Web 2.0 Solution” on page 271
- ▶ Chapter 14, “DLIModel Utility, DB Web Services, and XQuery” on page 293
- ▶ Chapter 15, “IMS Open Database and Universal Drivers” on page 321



IMS SOA enhancements

IMS is constantly improving its position as an SOA integration focal point, and we want to introduce the newest facilities that support this statement. We also refer you to some other IBM Redbooks publications, where previously announced features are described in detail:

- ▶ *IMS Connectivity in an On Demand Environment: A Practical Guide to IMS Connectivity*, SG24-6794
- ▶ *IBM IMS Version10 Implementation Guide A Technical Overview*, SG24-7526

In this chapter, we introduce you to:

- ▶ The enhancements that were introduced with IMS Version 10 that are associated with IMS SOA.
- ▶ The enhancements that were introduced after the general availability date for IMS Version 10 using the maintenance process.
- ▶ The enhancements that were introduced with IMS Version 11 that are associated with IMS SOA.
- ▶ Details about IMS application synchronous callout support.
- ▶ Details about OTMA enhancements.

8.1 On-demand and SOA-related IMS Version 10 features

Here are the on-demand and SOA-related features that were introduced with base IMS Version 10 and the associated SOA Integration Suite. Refer to *IBM IMS Version 10 Implementation Guide A Technical Overview*, SG24-7526 for details.

- ▶ Full XQuery support

Supported through the existing IMS JDBC interface, IMS XQuery support provides access using standard XQuery expressions to IMS full function data, which includes new IMS XML data.
- ▶ XML DB mapping enhancements

XML database mapping enhancements enable expanded mappings between new or existing IMS databases and visualized XML documents or collections to widen the scope of supported XML documents for new IMS databases and to ease disparate data integration across the enterprise.
- ▶ IMS SOA Composite Business Application support

IMS SOA Composite Business Application support is a new capability for IMS TM Resource Adapter clients that invokes IMS conversational transactions. Although the IMS TM Resource Adapter supported interactions with IMS conversational transactions, IMS Version 10 enhances this capability by allowing the iterations between a conversation to span multiple shareable sockets.
- ▶ XML adapter support for COBOL

Together with the IMS SOAP Gateway and IBM Rational Developer for z (RDz), using the XML Adapter support for COBOL you can reuse IMS applications as Web Services, which leverages open standards and utilizes flexible tooling support.
- ▶ IMS callout support

IMS callout support enables IMS applications as clients to inter-operate with business logic outside of the IMS environment. It provides for asynchronous callout to an external application (for example, a Web Service or a WebSphere application) through the IMS SOAP Gateway or IMS TM Resource Adapter.
- ▶ DLIModel utility enhancements

The DLIModel utility supports GSAM definitions now and generates appropriate metadata for consumption by the IMS JDBC driver and IMS DB resource adapter.
- ▶ MFS Web support

MFS Web support provides access to existing IMS MFS-based applications from IBM WebSphere Application Server environments to help protect your existing investments.
- ▶ IMS TM Resource Adapter PL/I application support

The IMS TM Resource Adapter provides PL/I support using Rational Application Developer tooling and the WebSphere Application Server to allow IMS PL/I applications to be enabled as Web Services without IMS application changes.
- ▶ APPC enhancements

LOCK and UNLOCK command support from both LU 6.2 devices and Open Transaction Manager Access (OTMA) clients.
- ▶ OTMA enhancements:
 - For any one OTMA client, you can activate message flood detection by specifying a maximum number of input messages that can be waiting concurrently for processing.

- OTMA automatically removes transaction pipes (Tpipes) after they are idle for three consecutive system checkpoints, if they are eligible for removal.
 - OTMA performs time-out processing for send-then-commit (CM1) messages, if an acknowledgement response is not received from an OTMA client within a period of time that the user specifies.
 - You can stop incoming transactions and commands from individual OTMA clients using the /STOP TMEMBER command without losing the connection or any other communications between the OTMA client and OTMA.
 - You can specify security levels for each OTMA client independently, which means greater flexibility in tailoring OTMA security.
 - The existing /DISPLAY commands for OTMA now return information for both commit-then-send (CM0) and send-then-commit (CM1) messages.
 - Support for the /LOCK and /UNLOCK commands is provided from OTMA clients and from Advanced Program-to-Program Communication (APPC) devices.
 - The OTMA descriptors are enhanced to allow you to define OTMA routing information. This enhancement eliminates the need to code OTMA routing exits to perform this task.
 - OTMA introduces a new optional flag in the OTMA prefix so that if you are using commit-then-send (CM0) transactions with multiple PURG calls, you can ignore the PURG calls to generate one output message with multiple segments.
 - OTMA implements the ACEE aging value capability to refresh user ID ACEEs. This value is checked when a user ID initiates a Resume Tpipe. If the OTMA aging value is exceeded, the user ID is considered expired and a RACROUTE is reissued for the Tpipe and user ID.
- ▶ **IMS Connect enhancements**
IMS Connect enhancements ease usability by providing support for Commit Mode 1 time out control, message flood control, alternate Client ID, and enforced client ID uniqueness. Security enhancements include support for mixed-case passwords, for password changes, and for an ACEE aging value. For more information, refer to 9.5, “What is new in IMS Connect Version 10” on page 175.
 - ▶ **IMS Java library support**
DB2 z stored procedures can use the IMS Java libraries to read data from IMS databases and return that data within a DB2 result set, back out to the caller of the DB2 stored procedure.

8.2 Post IMS Version 10 GA features introduced through maintenance

Due to the speed at which features for on-demand products must be delivered, IMS decided to supply some of these new enhancements through the service process rather than wait for the availability of a follow-on version. We provide the list of these features in Table 8-1 on page 148 by identifying the APAR, PTF, and maintenance level they were introduced by. If you recently installed a pre-maintained IMS Version 10 level, these features are probably already on your system.

Table 8-1 List of the IMS Version 10 maintenance available to enhance IMS SOA solutions

APAR / PTF / PUT LEVEL	Description of Feature
PK37758/UK23339/0703	IMS Connect adds two enhancements to the Send-Only protocol to ensure that Send-Only transaction inputs that are submitted on the same Tpipe are queued to the IMS message queues in the same order in which they were received by IMS Connect.
PK37905/UK23332/0703	Both the IMS Connect password Change (PWCH) and the PING requests are changed to not terminate the socket.
PK38197/UK27777/0708	OTMA provides a NOCHECK parameter option so that the /STA OTMA command does not need to be recovered during emergency restart. This reduces the chance of an OTMA message flood condition.
PK41554/UK25003/0705	Enhance OTMA Security Refresh for a single specific userID.
PK38720/UK31057/0711	The DFS555I message from a back-end Shared Queues system is queued and sent to the front-end IMS for the OTMA client.
PK61174	Allow IMS Connect Resume Tpipe requests to retrieve an ALTPCB message from a Shared Queues back-end system.
PK49317/UK41633/0811	Routing selected OTMA messages to the MTO and Auditlog besides the WTO systems console.
PK42286/UK25428/0705	IMS Connect compatibility enhancement in support of APAR PK43685 supplied by the IMS TM Resource Adapter to support it's 'CM0 No Wait ACK' enhancement.
PK73190	IMS Database Web Services is an extension to the DLIModel utility plug-in that provides a simplified way to expose IMS data as a Web service. You can use the DLIModel utility to generate an IMS application by using a standard IMS DL/I call that identifies the data you want to expose and deploy the generated IMS application to WebSphere Application Server for z/OS as an industry standard callable Web service.

8.3 Features introduced in IMS Version 11

The major On Demand associated features that are delivered with IMS Version 11 at the time of the publication of this book are:

- ▶ Open DataBase Adapter

The Open DataBase Adapter (ODBA) interface has a new command for IMS Version 11: CIMS CONNECT. This single command initializes the ODBA interface and connects to multiple IMS DB systems.

Prior to IMS Version 11, ODBA applications used the CIMS INIT command to initialize the ODBA interface and connect to a single IMS DB system. If the application wanted to connect to multiple IMS DB systems, it issued multiple CIMS INIT commands, one for each IMS DB system.

This new function is retrofitted to IMS Version 9 and IMS Version 10 systems by applying the following APARs:

- For IMS Version 9: PK66020 (PTF UK42176 on PUT 0812)
- For IMS Version 10: PK66022 (PTF UK42410 on PUT 0812)

- ▶ IMS Open Database support

Offers distributed access to IMS databases using industry-standard interfaces. Among the supported interfaces are JDBC, SQL, and the Java Connector Architecture for enterprise Java EE applications. Local access from various z/OS runtime environments are offered

too. This support extends the IMS Connect function as a gateway to IMS data, and adds to earlier support provided for IMS applications and operations. For more information, read Chapter 15, “IMS Open Database and Universal Drivers” on page 321.

► **IMS Connect enhancements**

These enhancements include configuration member support, enhanced commands, a cancel client ID option, improved availability with a new recorder trace facility, new and modified messages, and new event records for the HWSTECLO exit.

The enhancements to IMS Connect for IMS DB include a new DRDA-compliant application programming interface (API) and the ability to communicate with the new IMS Open Database Manager address space.

As part of the new Open Database enhancements, IMS Version 11 provides new Java drivers, called the IMS Universal drivers, that you can use to access your IMS data. IMS Connect supports the IMS Universal drivers with the DRDA API. Independent software vendors can also use any of the IMS Universal drivers to build packages that access IMS data.

Four new events (two single process events and two multiple process events) are added to the IMS Connect Event Recorder exit routine (HWSTECLO) in support of these Integrated IMS Connect enhancements for IMS DB.

For more information on many more IMS Connect enhancements, review Chapter 9, “IMS Connect and IMS Connect Extensions” on page 167.

► **Transaction expiration support**

IMS Version 11 can interrogate an expiration time that is associated with transactions from SNA and OTMA interfaces and discard (not process) based on an expiration time. This reduces processing costs and CPU cycles for the unwanted transactions.

IMS provides two levels of expiration specification:

- The transaction level that applies to all IMS messages and allows the expiration value to be defined through the following:
 - IMS System generation through a new attribute in TRANSACT macro.
 - Through the IMS DRD type-2 commands: CREATE TRANS or CREATE TRANDESC and UPDATE TRANS or UPDATE TRANDESC with the EXPRTIME set from 0 to 65535 seconds.
 - Through the Output Creation Exit Routine DFSINSX0.

Figure 8-1 on page 150 details the activity that occurs when the transaction level expiration time elapses.

Transaction Level Expiration

- **Transaction level expiration time**
 - A value, in seconds, that IMS uses to compare against the elapsed time of an unprocessed transaction input message
 - Values are compared when the scheduled dependent region retrieves the input message via GU

- **IMS actions for expired transaction message conditions:**
 - IMS ABENDU0243 issued without a storage dump created, dependent region controller reattached AND
 - Message DFS555I TRAN ttt ABEND ...
 - Displayed when an expired message is detected, OR
 - Message DFS2224I TRANSACTION ON A SQ BACK-END SYSTEM ABENDED
 - Displayed when an expired message is detected on a back-end shared queues system
 - 67D0 log record created

Figure 8-1 IMS activity when transaction level expiration times elapsed

The elapsed time that is used for the comparison includes the time when the message arrives in IMS (input timestamp) until the time when the message is scheduled into a dependent region and the region issues a GU to retrieve the message. If the input message is considered expired, it is deleted and the transaction is abended with a U0243 abend without the dump option selected. Additionally, depending on whether the environment is shared queues or not, either a DFS555I or DFS2224I message is sent to the end user.

- The second level expiration control is the message level specification that only applies to the OTMA environment. This support allows the message expiration time to be specified in the OTMA message prefix. Refer to “Open Transaction Manager Access enhancements in Version 11” on page 163 for more details on this enhancement.

The message types that are not part of the expiration time support are:

- IFP, MSC, conversational transactions, and switched transactions in a program to program switch.

▶ **OTMA enhancements**

We describe OTMA V11 enhancements in detail in 8.5.4, “Open Transaction Manager Access enhancements in Version 11” on page 163.

▶ **User exit enhancements**

Prior to IMS Version 11, when exit routines needed to be modified or brought online, IMS was stopped and restarted to recognize the changed or new exit routines. The User Exit enhancements solve this problem with new functions in IMS Version 11:

- A way to refresh exit routines online by using the new REFRESH USEREXIT command.
- A new way to query information about certain exit routines by using the new QUERY USEREXIT command.
- The ability to define multiple exit routines for a single exit type. In previous versions of IMS, you can only call a single exit routine for each exit type.

- Enhancements to the Standard User Exit Parameter List (SXPL).
- A new sub code that contains relevant data about the new exit routines is added to the IMS x'45' statistics log record.

This enhancement introduces or modifies these tasks:

- Specifying parameters of the DFSDFxxx PROCLIB member.
- Obtaining information about exit routines by using the QUERY USEREXIT command.
- Bringing a changed or new exit routine online by using the REFRESH USEREXIT command.
- Writing an exit routine.
- Diagnosing IMS problems (interpreting the information in the X'45' log record).

8.4 Introduction to IMS Callout support

When an IMS application program issues a callout request, IMS can be viewed as a client in a client-server relationship where the server is the external application to which IMS is making the callout request. If the application that initiates the callout request wants to continue processing and not wait for the response, the application must use asynchronous callout. If the application wants to wait for the response, the application must use synchronous callout.

IMS application callout support is not a new concept. There was callout support in various forms over the years, and we want to mention them before continuing our focus on the most current technologies. Methods to callout to other resources are:

- ▶ DL/I ISRT ALTPCB
- ▶ CPI-C/APPC calls
- ▶ TCP/IP sockets calls
- ▶ WebSphere MQ calls

Using diagrams and brief descriptions, we expand on the use of these methodologies before we discuss current callout facilities that are associated with TMRA and SOAP Gateway adapters and RYO solutions.

8.4.1 Using the DL/I ISRT ALTPCB call

Figure 8-2 on page 152 presents an example of a two-tiered IMS-managed request where the first IMS application, PGMA, issues an ISRT call to an ALTPCB that invokes PGM-B. PGM-B can be in the same IMS system, or Multiple Systems Coupling (MSC) can be used to access PGM-B in another IMS system. However each IMS program is an independent Unit of Work. The IMS DL/I API call provides for program-to-program asynchronous communications, where applications need to be aware of each other and manage the response correlation.

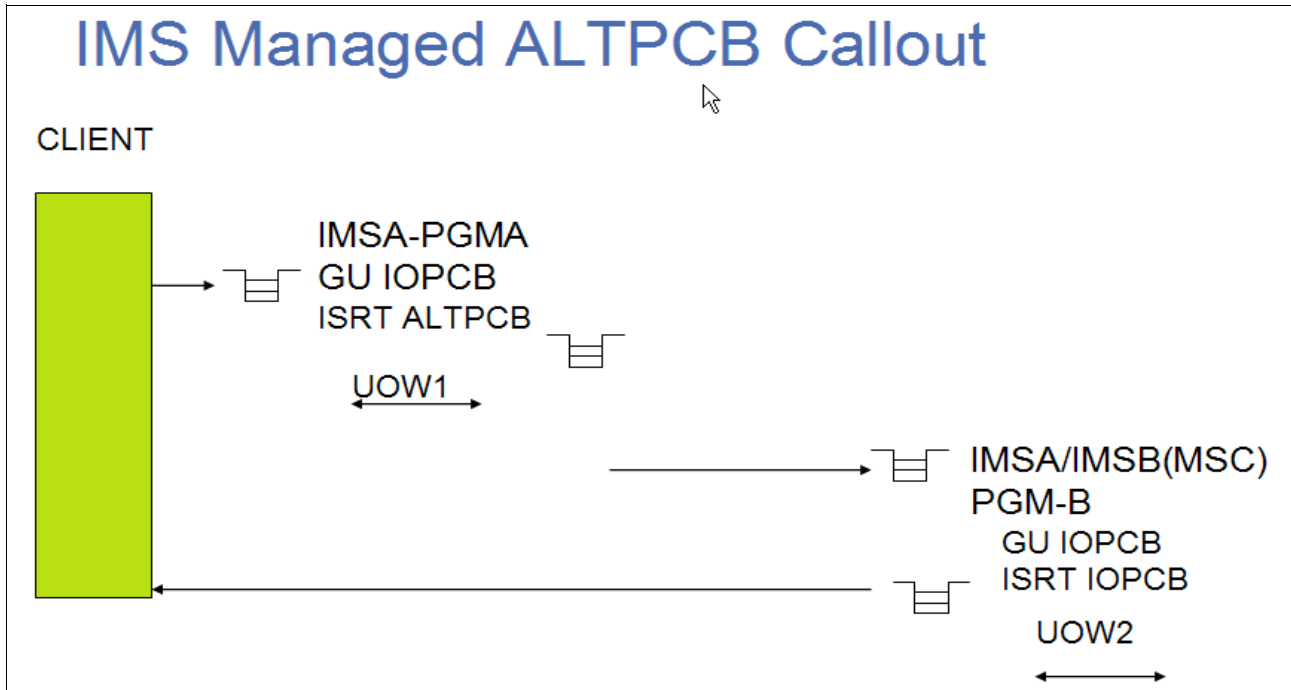


Figure 8-2 Callout through the use of the ALTPCB call

8.4.2 Using CPI-C/APPC calls

Figure 8-3 on page 153 presents an example of a two-tiered APPC conversation request where the conversation begins with PGMA on IMSA which invokes a conversation with TP-B on IMSA or IMSB (through MSC). PGMA on IMSA must wait for the remote application to return a response before the original program can reply to the client through an ISRT to the IOPCB.

When processing an explicit IMS APPC conversation with Common Programming Interface for Communications (CPIC), you can use the full set of CPIC command verbs. The IMS application is written specifically to cater only for APPC-triggered transactions. The standard IMS message queues are not used, and the IMS control region only helps create the APPC conversation directly between the APPC client and the IMS-dependent region to service this request.

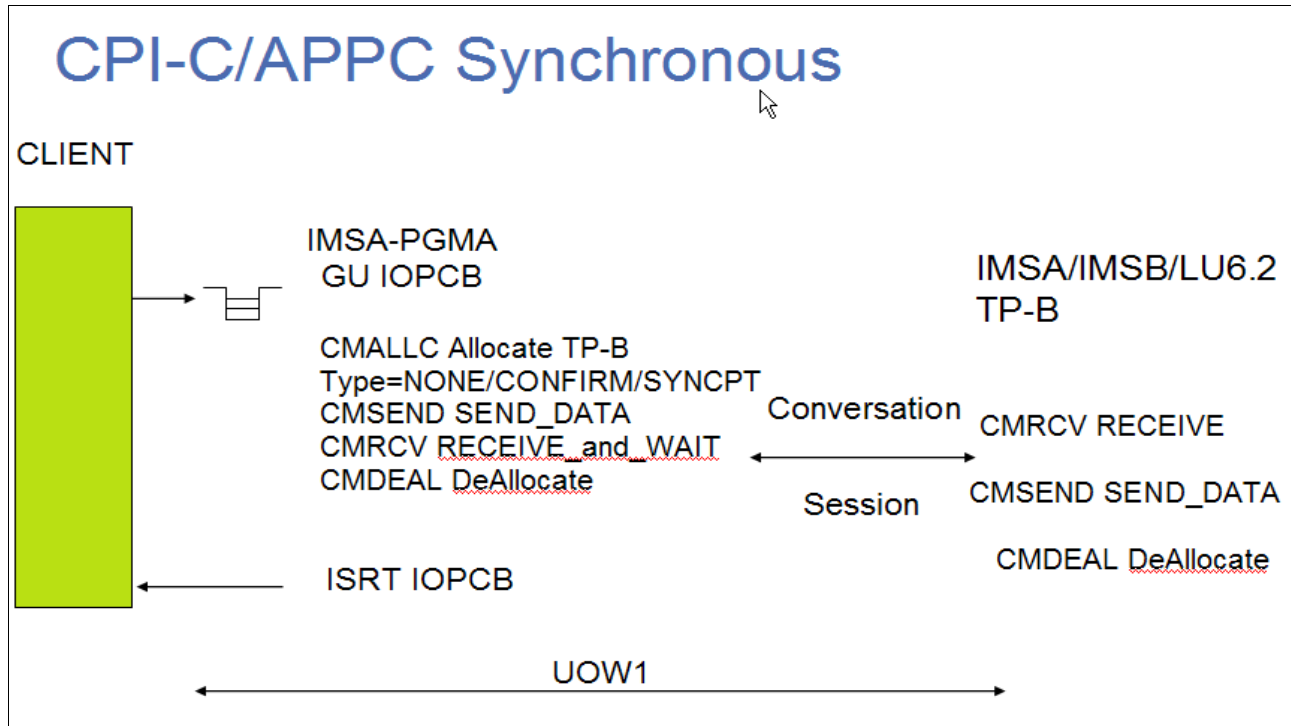


Figure 8-3 CPI-C / APPC synchronous callout flow

8.4.3 Use of TCP/IP sockets calls

Figure 8-4 on page 154 presents an example of a two-tiered TCP/IP connection where the socket connection begins with PGMA on IMSA, which invokes a connection with TCP/IP-B.

In this method of TCP/IP partner application-to-application callout communication, applications must be aware of others. IMS does not participate in coordinated commits.

There are a few terms mentioned in Figure 8-3 worth explaining:

- Datagram** A self-contained, independent entity of data that carries sufficient information to be routed from the source to the destination computer without reliance on earlier exchanges between this source and destination computer and the transporting network.
- Socket** An interface between an application process or thread and the TCP/IP protocol stack provided by the operating system.

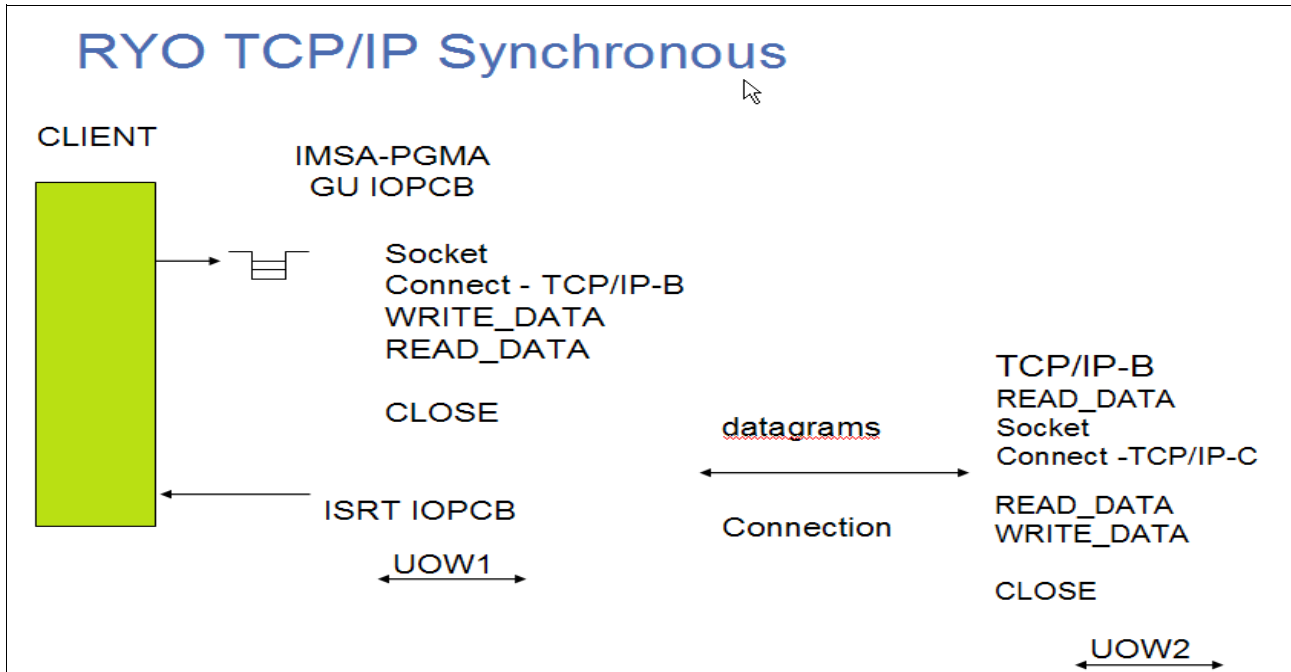


Figure 8-4 RYO application use of TCP/IP socket calls for callout

8.4.4 Using WebSphere MQ calls

Figure 8-5 presents a two-tiered MQ request where the request begins with PGMA in IMSA, invoking a MQ-B connection request.

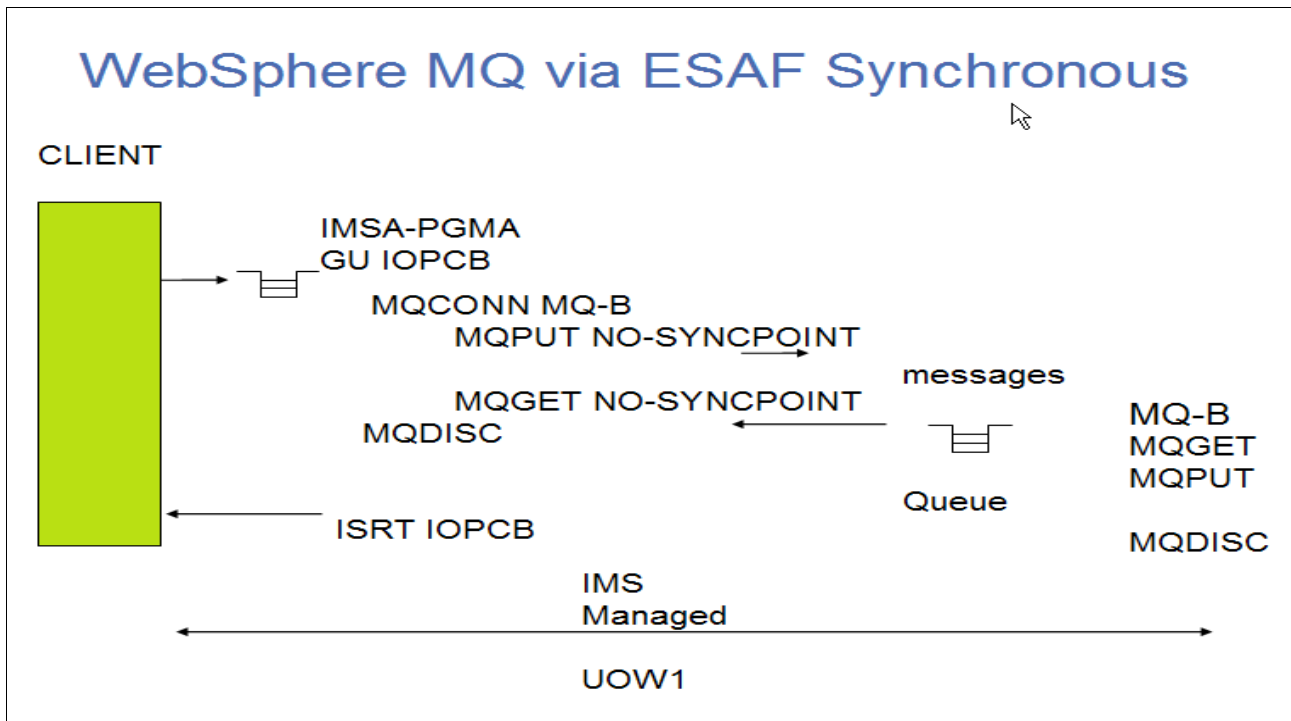


Figure 8-5 WebSphere MQ callout through the ESAF interface in synchronous mode

The IMS application does a GU to the IOPCB to retrieve the MQ trigger message but must then use the MQAPI (which removes the application from having to understand the communication protocols) to connect to and then put a message to MQ. The term *no-syncpoint* means that there is no coordinated syncpoint, and that the message will be allowed to flow. The reply is returned with an MQPUT to the MQ reply-to-queue. The IMS application must then close the queues and disconnect from MQ before returning a response to the client through an ISRT call to the IOPCB.

For more information about comparisons between IMS Connect and WebSphere MQ, refer to this Web site:

<http://w3-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100638>

In summary, these are a few of the methodologies that were utilized earlier to allow an IMS application to obtain data from either other IMS executed applications or non IMS sources. But continuing on this journey to position IMS as a consumer of Web Services, we introduce current adapters and application constructs.

8.4.5 Using TM Resource Adapter, SOAP Gateway, and RYO callout support

The three new approaches that you can now use to receive and process the callout request from an IMS application are:

- ▶ Using IMS TM Resource Adapter

Using IMS TM Resource Adapter you can create Java Platform Enterprise Edition (Java EE) applications to access IMS transactions over the Internet and to issue callout requests to external Java EE applications from IMS applications running in IMS dependent regions.

The IMS TM Resource Adapter includes a runtime component for the IBM WebSphere Application Server. Tooling support for the IMS TM resource adapter is available in IBM Rational Application Developer for WebSphere software and various Rational and WebSphere integrated development environments (IDEs).

- ▶ Using IMS SOAP Gateway

IMS SOAP Gateway is a Web Service solution to enable IMS applications as either Web Service providers or consumers. Tooling support for IMS SOAP Gateway is available in IBM Rational Developer for System z for generating the required Web Service artifacts based on connection and interaction information for communicating with IMS Connect and language structures in IMS applications.

IMS SOAP Gateway also provides a deployment utility to support IMS applications as either providers or consumers of Web Services.

- ▶ Writing your own IMS Connect TCP/IP client applications

You can write your own IMS Connect TCP/IP client applications and issue a RESUME Tpipe call to an OTMA routing destination, which is also known as a transaction pipe (or Tpipe), that is defined in an OTMA descriptor. This transaction pipe holds the callout requests. Your custom IMS Connect TCP/IP client application must poll the Tpipe to retrieve the callout requests.

8.4.6 Highlights of IMS synchronous callout support

The IMS synchronous callout support enables an IMS application to synchronously invoke external applications and receive the response back.

From your IMS application, you can invoke an external application and synchronously receive the response:

- ▶ A Java EE application, such as an Enterprise JavaBean (EJB) component, a message-driven bean (MDB), or a Web Service provider running in the WebSphere Application Server by using the IMS TM Resource Adapter.
- ▶ Other Web Service providers, such as Microsoft .NET or SAP XI, by using IMS SOAP Gateway.
- ▶ Any other applications, such as SAP applications or user-written applications, by using the IMS Connect interfaces.

Figure 8-6 presents highlights that are related to synchronous callout support.

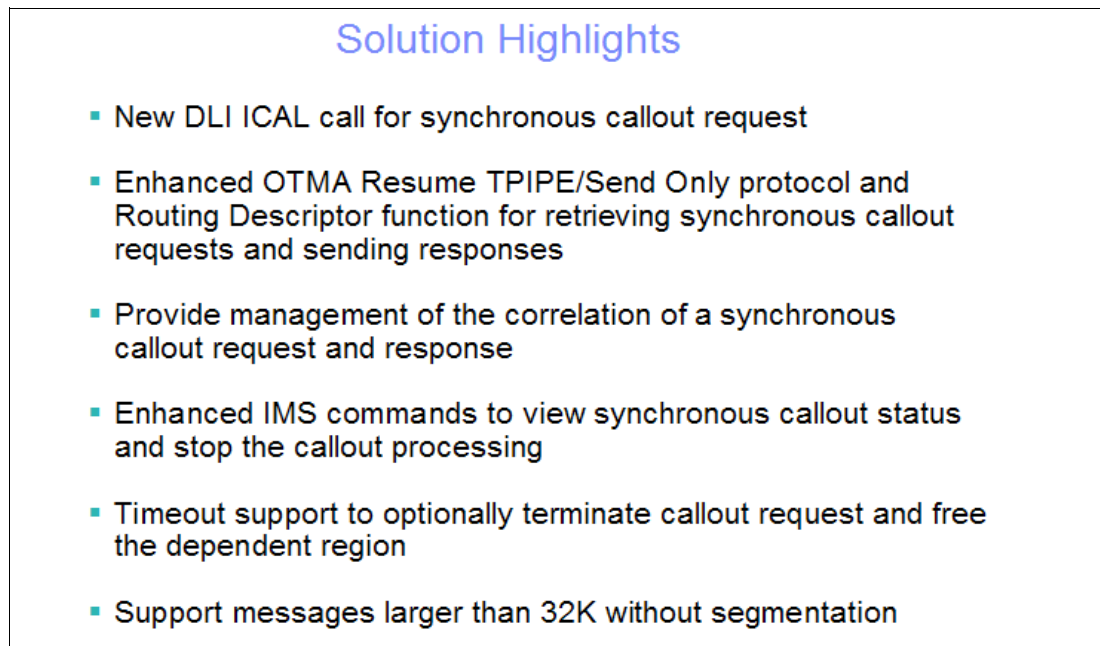


Figure 8-6 Feature highlights for synchronous callout support

To support IMS synchronous callout, new facilities are being developed.

A new DL/I call

To invoke an external application or Web Service, your IMS application must issue an ICAL and specify the OTMA descriptor name. Optionally, you can also specify a time-out value (the maximum time the IMS application waits for the response to return). To study the ICAL parameters in more detail, refer to 11.4.3, “Synchronous callout requests” on page 239.

Relief of 32K segmentation limitation

The 32K segmentation limitation for the input and output messages is removed, which allows IMS applications to send and receive large synchronous callout messages.

Concurrent processing

The synchronous callout function provides concurrent processing to ensure that the solution is robust and scalable. By using the enhanced RESUME Tpipe call and send-only protocol, you can have one connection for retrieving the callout requests and use other connections to return the response messages, simultaneously. IMS creates a correlation token to allow the response message to be correlated back to the correct IMS transaction instance.

Tooling support

Tooling support is provided to help generate code or configuration artifacts for enabling the synchronous callout function. You can use:

- ▶ IBM Rational Developer for System z tooling to generate the required XML converters and callout message correlators from COBOL or PL/I source files when you use IMS SOAP Gateway.
- ▶ Rational Application Developer for WebSphere Software to create MDB code, or you can write your own Java code to consume callout messages from IMS applications when you use the IMS TM Resource Adapter.

Figure 8-7 presents the message flow of the synchronous callout function. After issuing the ICAL, the application must wait for a response from a potential variety of sources based on the connectivity options selected.

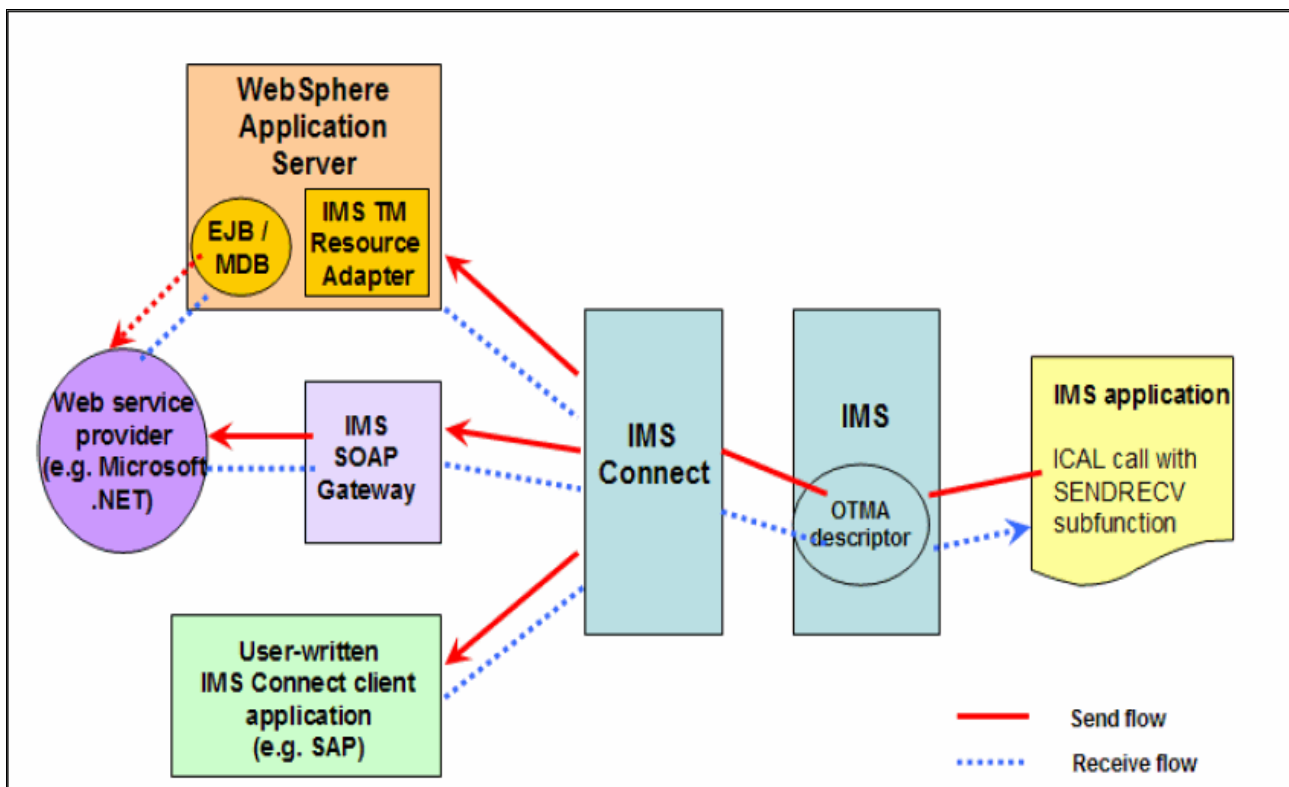


Figure 8-7 Synchronous callout support message flow

8.4.7 Stages to implement the synchronous call function

Figure 8-8 on page 158 presents the general task flow for implementing and deploying your synchronous callout application. We describe the use of the ICAL and then define the OTMA descriptor. Your three choices to implement this call out function are to:

- ▶ Use the TMRA V10.3
- ▶ Use the SOAP Gateway V10.1
- ▶ Use your own IMS Connect Client

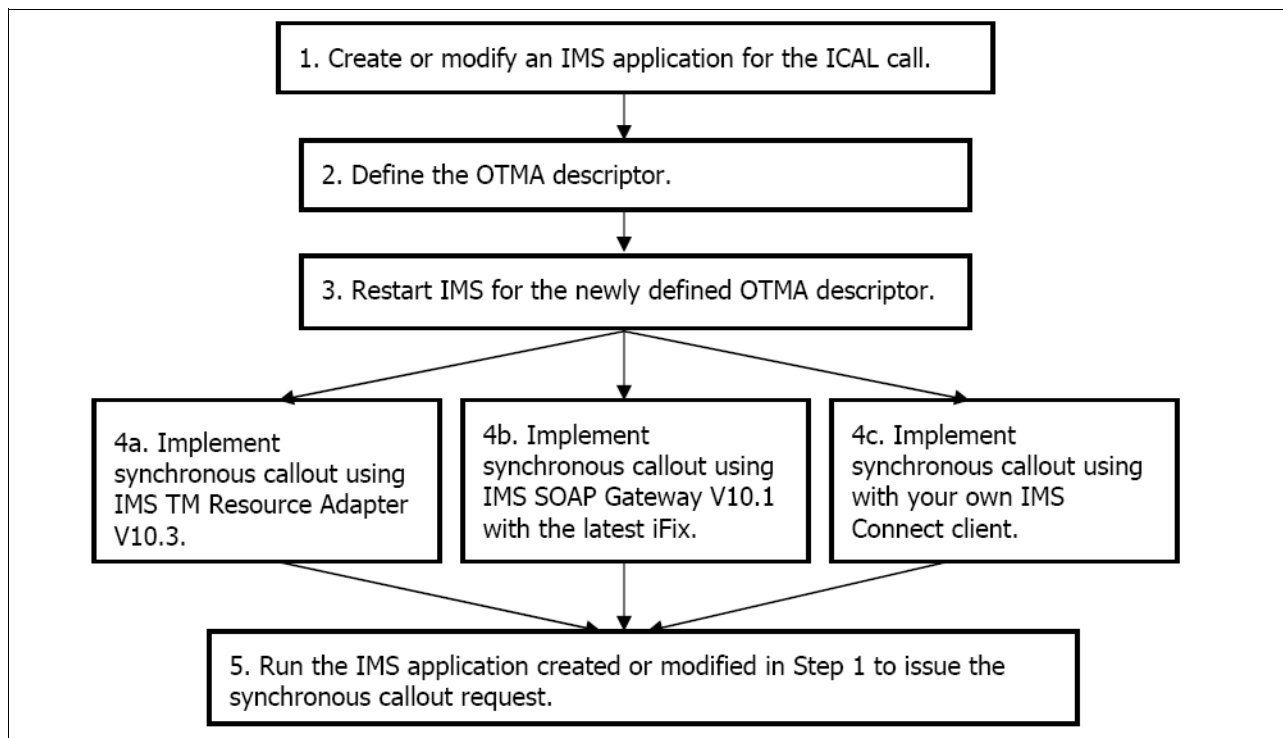


Figure 8-8 Task flow diagram for implementing the synchronous callout function

8.4.8 Creating or modifying your IMS application to issue an ICAL call

Use the IMS ICAL call with the AIBTDLI interface to issue a synchronous callout request from your IMS application. The ICAL call enables an IMS application program that runs in the IMS TM environment to send a synchronous request for data or services to a non-IMS application program or service. The syntax is:

```
>>-ICAL--aib--request_area--response_area-----<<
```

The parameter aib specifies the application interface block (AIB) that is used for the call.

The 'request_area' contains the request message data that is sent from the IMS application program to the application that is specified in the OTMA descriptor. The AIBOALEN field specifies the length of the request message data. Because the ICAL call bypasses IMS TM message queuing, the format of the request area does not require the LLZZ fields.

The 'response_area' specifies the response area to use for this call. This response area must be large enough to hold the response returned. If the response area is not large enough to contain all of the returned data, IMS returns partial data. When partial data is returned, the AIBOAUSE field contains the length of the returned data in the response area, and AIBOALEN contains the actual length of the response message.

8.4.9 Defining the OTMA descriptor for destination routing

OTMA client descriptors are built during IMS initialization. The descriptors are included in DFSYDTx members of IMS.PROCLIB.

Use the optional keyword SYNTIMER for OTMA descriptors to specify the time in hundredths of a second to wait for synchronous callout processing to complete. The range is from 0 to

999999. The AIBRSFLD parameter in the ICAL SENDRECV call can override this value. If no time out value is specified in either the AIBRSFLD parameter of an ICAL SENDRECV call, or in the OTMA descriptor by using the SYNTIMER keyword, the default is 10 seconds before synchronous call processing if the dependent region times out. If a value larger than 999999 is specified, the maximum value is used.

Example 8-1 shows an OTMA descriptor for a destination named OTMDEST1 that routes messages to the TMEMBER named HWS2 and a Tpipe named HWS2SOAP with a time-out value of 5 seconds.

Example 8-1 OTMA descriptor using a time-out value of 5 seconds

```
D OTMDEST1 TYPE=IMSCON TMEMBER=HWS2 Tpipe=HWS2SOAP SYNTIMER=500
```

The optional SYNTIMER parameter is not valid if the descriptor is of TYPE=NONOTMA.

8.4.10 Software requirements for synchronous callout support

The software requirements for the synchronous callout function are:

- ▶ Maintenance is required for IMS Version 10 and some IMS Integration Suite products:
 - APAR PK70330 included in PTF UK40215 for the IMS Connect component of IMS Version 10
 - APAR PK70078 included in PTFs UK40363 and UK40364 and PK73224 included in PTF UK40813 for IMS Version 10
 - Activation APAR PK71135 included in PTF UK42415 for IMS
 - Activation APAR PK74168 included in PTF UK42459 for IMS Connect
 - PK75824 for synchronous callout IVP sample
 - PK75209 JBP and JMP support for issuing callout requests. Apply this APAR if you are issuing callout requests from Java applications that run in a JBP or JMP region.
 - PK75460 for IMS TM Resource Adapter support for synchronous callout support. Apply this APAR if you are using the IMS TM Resource Adapter for synchronous callout processing.
- ▶ IMS and IMS Connect Version 10: The target of the synchronous callout request must be WebSphere Application Server (through IMS TM Resource Adapter), IMS SOAP Gateway, or a user-written client where IMS Connect is used as the TCP/IP interface.
- ▶ For IMS TM Resource Adapter, the requirements are:
 - IMS TM Resource Adapter Version 10.3, which includes a runtime component for WebSphere Application Server and a development component for IBM Rational or WebSphere development environments
 - A Rational or WebSphere development environment that includes IBM Rational Application Developer for WebSphere Software Version 7 with the latest fix pack (Version 7.0.0.4 is the minimum level. Version 7.5 is recommended)
 - WebSphere Application Server Version 6.1 with the latest fix pack
- ▶ For IMS SOAP Gateway, the requirements are:
 - IMS SOAP Gateway Version v10.1 +iFIX2
 - Rational Developer for System z Version 7.1.1 or later

8.4.11 Synchronous callout restrictions

The restrictions for synchronous callout functions are:

- ▶ Two-phase commit is not supported.
- ▶ In a shared queue environment, the response message of a synchronous callout request must flow back to the originating IMS to be processed correctly.
- ▶ The synchronous callout design is provided for IMS TM users. BMP or JBP applications running in a DBCTL environment are not supported.
- ▶ The callout processing inside the IMS application must be single-threaded, that is, the IMS application can only issue one synchronous callout call at a time. It must wait for the response (or time-out) before it can issue the next callout request.
- ▶ For IMS SOAP Gateway, the maximum XML message size for both the request and response for synchronous callout messages is 32 MB for COBOL applications and 16 MB for PL/I applications.

In the next section, we discuss OTMA because it is a component of the IMS core product, and it plays an important role in SOA solutioning.

8.5 Open Transaction Manager Access overview

The four main communication protocols that are used in IMS environments are:

- ▶ TCP/IP between the distributed participants and IMS Connect
- ▶ OTMA between IMS Connect and IMS
- ▶ SCI between IMS Connect and IMS for Open database support
- ▶ Traditional VTAM interfaces

The OTMA implementation is specific to IMS in a z/OS environment. The domain of the protocol is restricted to the domain of the z/OS Cross-System Coupling Facility (XCF) transport layer. OTMA is the connection protocol that is used between the IMS connector and IMS. Also OTMA is the session and transport layer of the network protocol. OTMA, however, does not exactly conform to the OSI model because OTMA can process several sessions simultaneously using a single transport connection.

OTMA performs some of the basic functions of the OSI transport layer (those not performed by XCF), so it is simplest to think of OTMA as a combined session and transport layer with the transport layer comprised of both XCF and OTMA. OTMA is designed to be a high-performance, comprehensive protocol that allows z/OS programs to access IMS applications.

The means that OTMA is used are:

- ▶ OTMA Callable Interface

The IMS OTMA Callable Interface (C/I) provides a high-level interface that allows application programs on other z/OS subsystems to access IMS applications through OTMA. The interface consists of API calls that are available to a C/C++ program. The API calls are used to join the IMS/OTMA XCF group to connect to IMS, to allocate communication sessions, to send IMS transactions and commands, to receive output from IMS, to close communication sessions, and to leave the XCF group.

For more documentation about this callable interface, refer to *IMS V10 Communications and Connections Guide*, SC18-9703.

► WebSphere MQ

WebSphere MQ on z/OS through the IMS bridge is an OTMA client. The IMS adapter is the interface between IMS application programs and the WebSphere MQ subsystem. It makes it possible for IMS application programs to use the MQI (message queue interface). IMS transactions can also be scheduled through WebSphere MQ queuing drivers.

The IMS adapter receives and interprets requests for access to WebSphere MQ using the External Subsystem Attach Facility (ESAF) that IMS provides.

The complexity of OTMA is completely hidden from you, and you are only presented with the WebSphere MQ API (putting messages in queues and reading messages from queues).

For more documentation about this aspect, refer to *WebSphere MQ for z/OS System Setup Guide (V6)*, SC34-6583.

► IMS Connect

IMS Connect is an IMS Open Transaction Manager Access client. After configuring OTMA, the interactions from clients are steered by the frames over the TCP/IP connections. The required interaction models with IMS are:

- SYNC_SEND
- SYNC_SEND_RECEIVE
- SYNC_END_CONVERSATION
- SYNC_RECEIVE_ASYNCOUTPUT

8.5.1 Activating Open Transaction Manager Access

OTMA is based on XCF. Each partner in an XCF-based protocol must belong to the same XCF group and must have a member name.

To enable IMS to use OTMA, specify the XCF group name and IMS OTMA member name during system definition. To start OTMA, you can use the OTMA=Y startup parameter in the IMS procedure during IMS system definition or after an IMS restart issue the type-1 command /START OTMA.

8.5.2 Open Transaction Manager Access transaction pipe

The key to message flow for OTMA is the transaction pipe, which is the logical connection between the server and the OTMA client. An OTMA client includes the transaction-pipe name in the message-control information section of the message prefix for the input message. IMS then associates application output for an OTMA client with a specific transaction pipe.

A transaction pipe is analogous to an IMS logical terminal (LTERM). For each LTERM, IMS maintains a connection between the queue and the physical node that receives the output. OTMA does not use a LTERM but still must maintain a connection between the client and IMS. This connection is the transaction pipe, or Tpipe.

IMS uses the transaction-pipe name to associate all input and output with a particular client. The association between the transaction output and its ultimate destination (for example, a user at a terminal or a printer) is not made within IMS (as is the case with LTERM's), but is your responsibility.

Figure 8-9 on page 162 illustrates how transaction pipes fit in an OTMA client/server environment. As shown, transaction pipe structures reside in the OTMA layer only for the server. XCF, which resides in the transport layer, can be thought of as an inter-process

communication layer because it provides communication between the client process and the server process. In the context of this description, the XCF client is IMS Connect.

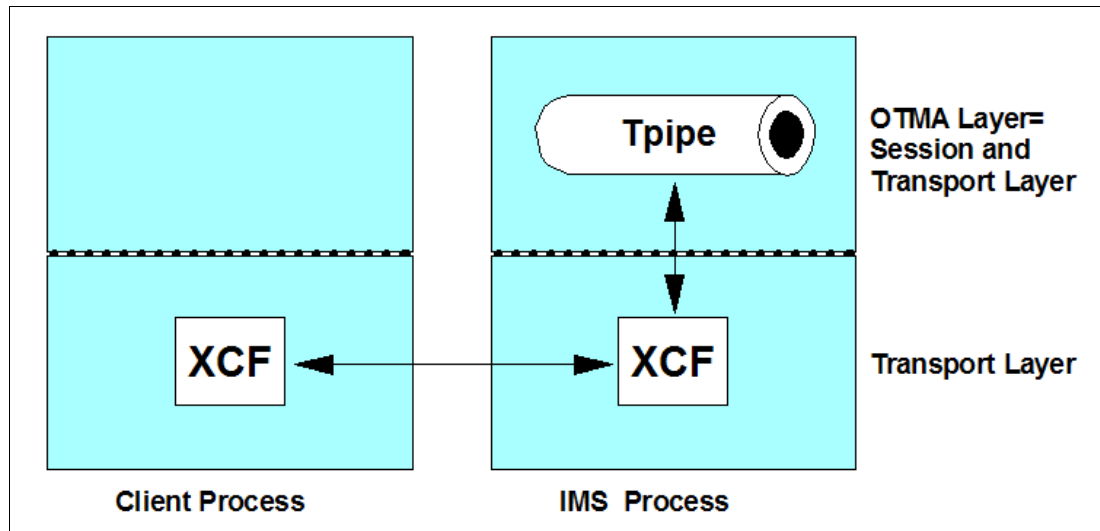


Figure 8-9 How transaction pipes fit in an OTMA client/server environment

8.5.3 Open Transaction Manager Access enhancements in Version 10

The ways that OTMA is enhanced for IMS Version 10 are:

- ▶ You can define a limit to the number of input messages from any one OTMA client that can be waiting at the same time to be processed. A message flood condition occurs when too many transactions are waiting to be processed by OTMA. A message flood condition can use up all available LSQA storage and result in a z/OS S40D abend. This condition might be caused by either an abnormal increase in incoming messages or a problem that prevents IMS from processing the messages normally.

OTMA issues warning messages as the message count approaches the limit. When the message count reaches the limit, OTMA identifies a message flood condition and rejects any new messages from the OTMA clients until the message count drops or a new limit is set.

The message flood control enhancement in IMS Version 10 is automatically enabled with a default limit of 5000 messages. The messages for the input message flood condition that are routed to the WTO system console are:

- DFS1988W OTMA input messages from member yyyy have reached xx% of the max concurrent input message limit zzzz.
- DFS1989E OTMA input messages from member yyyy have reached the maximum concurrent input message limit zzzz.
- DFS0767I OTMA message flood condition was relieved for member yyyy.

To provide compatibility with previous releases and deactivate the support, either specify an INPUT value of zero in a descriptor or issue the /STA TMEMBER INPUT command.

- ▶ OTMA removes eligible transaction pipes automatically if they are idle for three consecutive system checkpoints.
- ▶ To free dependent regions that are waiting for an ACK or NAK response, OTMA now performs time-out processing for send-then-commit (CMO) messages if the ACK or NAK response is not received from an OTMA client within the time period that you specify. The

OTMA client must be using an appropriate sync level value. After the ACK or NAK response times out, the dependent region is released to process other transactions.

- ▶ You can stop incoming transactions and commands from individual OTMA clients. The connection remains active, and transactions that are already being processed are unaffected. Conversational transactions that are already in progress can complete normally, and OTMA still processes all of the OTMA protocol commands.
- ▶ You can specify security levels for each OTMA client independently, which provides much greater flexibility in tailoring OTMA security.
- ▶ The /DISPLAY TMEMBER TPIPE command displays information for both commit-then-send and send-then-commit messages, including the accumulated input message count and the name of the Destination Resolution (DRU) exit routine.
- ▶ A CM1 (Send-then-Commit) timer was introduced, which addresses output reply messages that OTMA sends for Synclevel=confirm or synclevel=syncpt processing. When IMS waits in “wait-syncpoint” or “wait-RRS” status, locks are held, and the dependent region is occupied. A default time-out value of 120 seconds for ACK/NAK not received was introduced, which protects the IMS environment from remote applications that do not respond in a timely fashion or network delays

To provide compatibility with previous releases and to deactivate the support, specify a value of zero in a descriptor or issue the /STA TMEMBER TIMEOUT command.

8.5.4 Open Transaction Manager Access enhancements in Version 11

The IMS Version 11 OTMA functionality enhancements are:

- ▶ Consistency enhancements for Shared Queues environments

Both error message handling and super member support were enhanced:

- Shared Queues error message handling

Before this support, OTMA asynchronous support for Shared Queues environments documented a restriction relating to a back-end Shared Queues system abend situation. Transaction abends on a back-end system did not support sending the DFS555I message to the waiting remote client attached to a front-end IMS. The remote clients implemented a time-out flow to ensure that the connection would eventually be broken.

IMS Version 11, IMS Version 9 with maintenance PK35745 through PTF UK31054, and IMS Version 10 with PK38720 through PTF UK31057, lifted this restriction and allows the DFS555I message to flow from the back-end IMS through the front-end to the waiting remote client application. This provides consistency for IMS shared queues environments so that the result of the abend in either front-end IMS or back-end IMS is the same.

- Enhancement to OTMA super member support

Before this enhancement, when an ALTPCB output message was generated on one of the back-end SQ systems, super member support only allowed retrieval of messages that were already on the queue. This did not support creating a program that can automatically wait for new messages.

The OTMA super member support for Shared Queues provides the capability for an IMS Connect client to connect to any front-end IMS and retrieve an ALTPCB message that is created or will be created in any of the back-end systems in the shared queues group. The ability, using super member support to create a listening remote client that can wait for and retrieve these types of messages as they are queued, was previously documented as a restriction.

This support is available for IMS Version 9 through maintenance PK56730 with PTF UK35483, and with APAR PK61174 for IMS Version 10.

▶ Message level input message time-out support:

The OTMA environment supports two levels of specifying when an unprocessed input message must expire: the transaction level specification and enhanced support for message level specification. We already discussed transaction level expiration support in 8.3, “Features introduced in IMS Version 11” on page 148

Message level specification for the expiration value is supported only for OTMA interactions. When specified in the OTMA prefix, the expiration value overrides any other value that was provided through the transaction level specification options. OTMA supports two formats for message level expiration times: a STCK format and an elapsed time format. OTMA clients can choose to use one or the other format for individual messages.

- ▶ The new CM0 ACK time-out support allows you to detect a hang condition in an OTMA transaction pipe and reroute commit-then-send (CM0) transactions to a time-out message queue so that the remainder of the I/O PCB output on the transaction pipe can continue to flow normally.
- ▶ The new transaction monitoring enhancements allow you to monitor OTMA resources to protect IMS from flooding with excessive messages, expire unwanted OTMA input transactions, and send OTMA protocol or heartbeat messages with resource information.
- ▶ The new OTMA type-2 commands:

- Gives you the ability to monitor the workload in OTMA, specifically the messages in the OTMA send-then-commit (CM1) message queue. The new QUERY OTMATI command allows the user to monitor the workload in IMS OTMA, specifically the “transaction instances” or messages in the queue that are received and sent through OTMA (internally represented by transaction instance block, TIB). Monitoring the TIB data provides a mechanism to diagnose potential problems that might arise and, correspondingly to respond pro-actively in an attempt to avoid or circumvent such problems. Examples of a problem in this area would be excessive storage usage by control blocks associated with each transaction instance.

When the command is issued without parameters, e.g., QUERY OTMATI, the information displays the TMEMBER, the Tpipe under the TMEMBER, and the total number of Transaction Instance control blocks which might represent queued messages, continuing IMS conversations, or other problems.

- Allows you to make dynamic modifications to OTMA descriptor entries through DRD commands, which eliminates the need to restart IMS if a destination routing definition is changed.

▶ Enhancements to route messages to the MTO/Auditlog

The input message flood condition messages that are listed in 8.5.3, “Open Transaction Manager Access enhancements in Version 10” on page 162 are routed to the WTO system console, and also to the MTO and Auditlog so that an AOI exit can be written to monitor these messages and associated resource conditions.

This support is also available through IMS Version 9 maintenance PK47987 with PTF UK28262, and with PK49317 with PTF UK41633 in Version 10.

8.5.5 Open Transaction Manager Access descriptors

OTMA descriptor IMS Version 10 enhancements allow the specification of additional properties based on clientID or alternate (ALTPCB) destinations.

The OTMA Destination Resolution exit routine (DFSYPX0) determines whether an asynchronous output message needs to be routed to an OTMA destination or a non-OTMA destination. If the message must be routed to an OTMA destination, the OTMA Destination Resolution exit routine can then determine the final OTMA destination client or Tpipe.

You can use the OTMA descriptor to avoid coding this user exit.

OTMA provides two types of descriptors:

► An OTMA client descriptor

The OTMA client descriptor specifies certain characteristics of an OTMA client, such as its DFSYDRU0 exit routine, its send-then-commit message time out values, or its message flood threshold value. Use OTMA client descriptors to provide information about a specific OTMA client to IMS.

OTMA client descriptor entries are identified in the DFSYDTx PROCLIB member by an M in column one of the descriptor entry. Using an OTMA client descriptor, the attributes of an OTMA client that you can specify to OTMA and IMS are:

- The name of the OTMA Destination Resolution exit routine that the OTMA client uses.
- For send-then-commit messages that use synclevel=confirm or synclevel=syncpt, the time-out value that OTMA must use when waiting for an ACK or NAK response from the OTMA client.
- The maximum number of input messages from the OTMA client that can be processing in an IMS system before triggering a message flood condition.

► An OTMA ALTPCB descriptor:

The OTMA ALTPCB destination descriptor specifies a destination for OTMA ALTPCB output sent to either IMS Connect or non-OTMA destinations such as an SNA terminal or printer.

The OTMA ALTPCB destination descriptor provides a simpler method of describing ALTPCB destinations than the method that the OTMA Destination Resolution exit routine (DFSYPX0) and the OTMA user data formatting exit routine (DFSYDRU0) provide.

The OTMA ALTPCB destination descriptor also supports the routing of callout requests to services connected to by the IMS TM Resource Adapter and IMS SOAP Gateway.

OTMA ALTPCB destination descriptors are coded in the DFSYDTx PROCLIB member and are identified by a D in column one of the descriptor entry as illustrated in Example 8-2.

The ALTPCB destination attributes that you can specify are:

- The destination type: The OTMA ALTPCB destination descriptor supports only IMS Connect, tagged by “IMSCON”, and non-OTMA destinations, tagged by “NONOTMA”, as destination types.

For IMS Connect destinations: An OTMA tmember name or an OTMA super member name and an optional Tpipe name.

For IMS Connect XML conversion support for IMS SOAP Gateway: a XML adapter name and a XML converter name.

When specifying a destination name in an OTMA ALTPCB descriptor, you can generalize destination names by using an asterisk as a wildcard character or as a mask of the final characters in the destination name field.

Example 8-2 DFSYDTx

```
M HWSICON1 DRU=DFSYDRU0
D OTMACL99 TYPE=IMSCON TMEMBER=HWS1 Tpipe=HWS1TP01
```

```
D OTMACL*  TYPE=IMSCON TMEMBER=HWS2
D PRNTR3A  TYPE=NONOTMA
D SOAPGWAY TYPE=IMSCON TMEMBER=HWS2 Tpipe=HWS2SOAP
D SOAPGWAY ADAPTER=XMLADPTR CONVERTR=XMLCNVTR
```

Example 8-2 shows a list of OTMA descriptors in the DFSYDTx PROCLIB member:

- ▶ The first is a TMEMBER descriptor specifying the DFSYDRU0 exit for TMEMBER HWSICON1.
- ▶ The second descriptor is a destination routing descriptor for destination OTMACL99 to route messages to TMEMBER HWS1 and Tpipe HWS1TP01.
- ▶ The third is a destination routing descriptor for destinations that match the mask OTMACL*. They are routed to IMS Connect TMEMBER HWS2, with a Tpipe of the destination that matches the mask (for example, OTMACL04).
- ▶ The fourth descriptor is another destination routing descriptor for destination PRNTR3A that is routed to IMS.

The last descriptor (SOAPGWAY) is a destination routing descriptor for IMS SOAP Gateway. It is routed to IMS Connect TMEMBER HWS2 with Tpipe HWS2SOAP and underwent XML transformation through an adapter and converter code.

OTMA descriptors can be used with the IMS Callout functions and should prevail over the coding of the exit's.

This concludes our discussion of IMS SOA enhancements.



IMS Connect and IMS Connect Extensions

IMS Connect is a gateway that operates in an address space on the z/OS platform between a service consumer (client) and IMS, if inbound, or IMS and a service producer (server), if outbound.

The support that IMS Connect provides is:

- ▶ Provides commands to manage the communication environment.
- ▶ Assists with workload balancing.
- ▶ Reduces design and coding efforts for client applications.
- ▶ Offers easier e-business access to IMS applications and operations with advanced security and transactional integrity.

IMS Connect Extensions improve upon your investment in IMS and IMS Connect. The support that it provides is:

- ▶ Provides the ability to analyze problems and optimize performance by recording key IMS Connect events.
- ▶ Improves system security with flexible access control.
- ▶ Creates detailed IMS Connect reports in conjunction with IBM's IMS Performance Analyzer for z/OS.
- ▶ Assists in the resolution of IMS Connect problems in conjunction with IBM's IMS Problem Investigator for z/OS.
- ▶ Allows you to view graphical real-time reports of TCP/IP activity with IBM's OMEGAMON XE for IMS on z/OS.

In this chapter, we first discuss IMS Connect and then IMS Connect Extensions.

9.1 IMS Connect and SOA

As you can see in Figure 9-1, the service consumer can connect through TCP/IP (local connect is also available) to the IMS server over the IMS Connect gateway. The local support facility allows communication using Program Calls (PC) without requiring TCP/IP from a Web serving application to IMS in a z/OS or OS/390 environment, which eases the management in this environment.

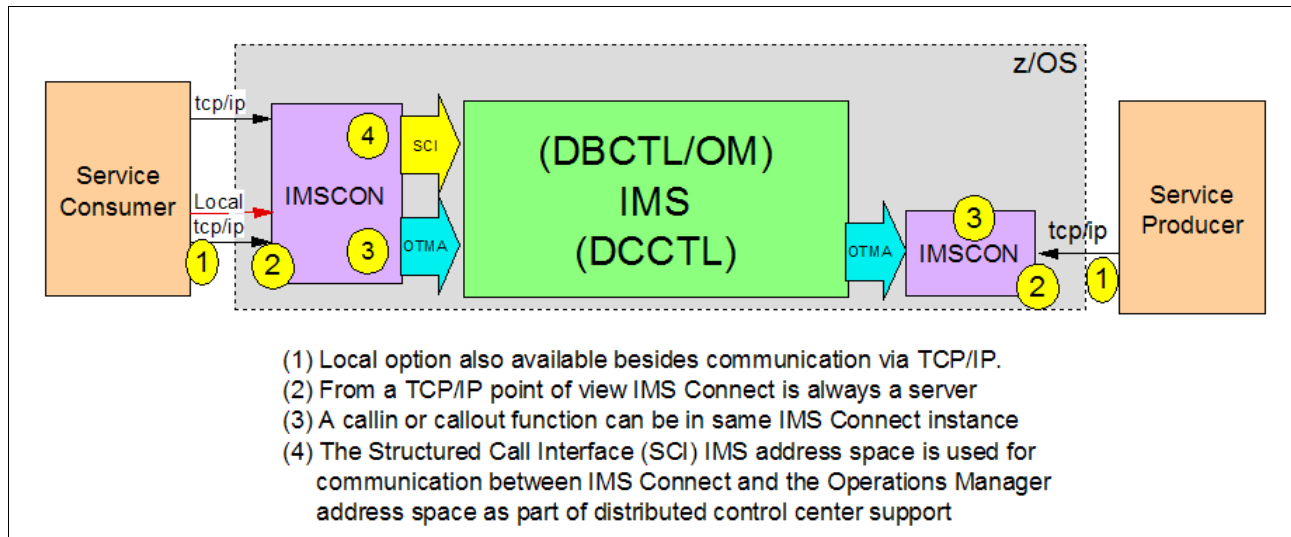


Figure 9-1 IMS Connect in a Service Oriented Architecture

The IMS Connect server runs on System z, although it can be invoked from clients on either distributed or System z platforms. Although the IMS Connect function is shown twice in Figure 9-1, both functionalities of call-in or call-out can be combined in one instance.

Traditionally IMS Connect is used as a gateway to reach the DCCTL (communication portion) of IMS. Since IMS Version 9, it is also used as a communication path for the IMS operation management (OM) component for distributed IMS single point of control as provided from a distributed control center to operate IMS within an IMSPLEX. The Structured Call Interface (SCI) component of IMS is used for the IMS Connect to OM communication. In 9.11.2, "IMS Connect support for IMSplex and the IMS Control Center" on page 200 we indicate how this SCI protocol is used.

Figure 9-2 on page 169 shows the manner that IMS subsystems can utilize IMS Connect.

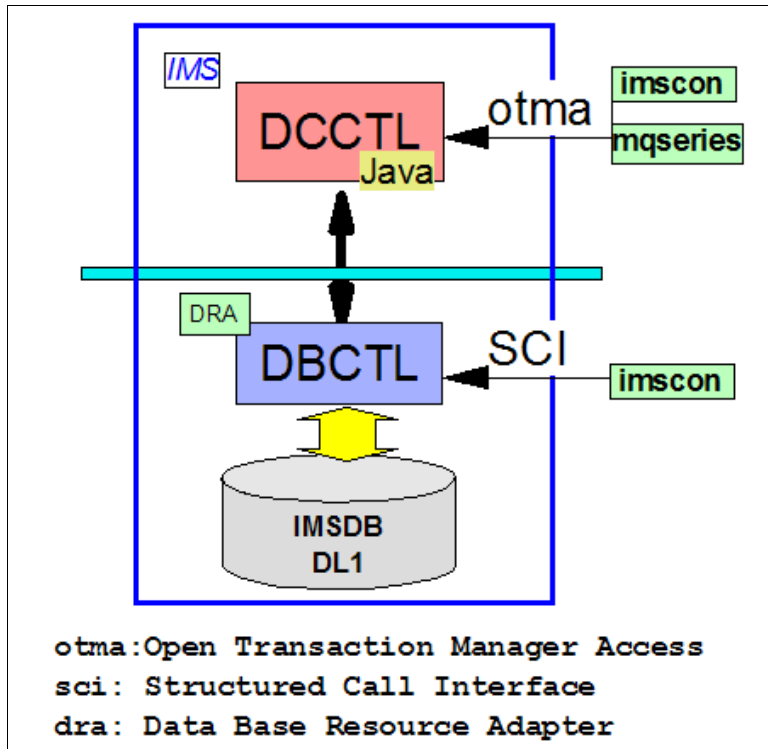


Figure 9-2 IMS Connect interfaces to IMS subsystems

Our objective in this chapter is to explain the circumstances in which you can use IMS Connect, who can be the consumer or the producer, what are the considerations for the implementations, and what are the building blocks. Details about application APIs and frame content during the exchanges between the components and wizards are at:

- ▶ *Powering SOA with IBM Data Servers*, SG24-7259
- ▶ *IMS Connectivity in an On Demand Environment: A Practical Guide to IMS Connectivity*, SG24-6794
- ▶ *IMS V10 Communications and Connections Guide*, SC18-9703
- ▶ *IMS V10 System Definition Guide*, GC18-9998

9.2 TCP/IP

By examining Figure 9-3 on page 171, you can see that TCP/IP is used between IMS Connect and external partners. These partners can be located on distributed systems or within the same z/OS image. If the partners exist within the same z/OS image as IMS Connect, PC calls can be used for communication; however, using HiperSockets, TCP/IP can also be utilized.

Mainframe HiperSockets is a technology that provides high-speed TCP/IP connectivity within a central processor complex. It eliminates the need for any physical cabling or external networking connection between servers that are running in different LPARs. The communication is through the system memory of the processor, so servers are connected to form an internal LAN. The HiperSockets implementation is based on the OSA-Express Queued Direct I/O (QDIO) protocol, hence HiperSockets is also called internal QDIO, or IQDIO. The microcode emulates the link control layer of an OSA-Express QDIO interface.

If you want to read more about HiperSockets, refer to:

HiperSockets Implementation Guide, SG24-6816

Other communication options influence the effectiveness of the TCP/IP communication. We discuss a few elements here, but a more complete explanation is in:

IMS V10 Communications and Connections Guide, SC18-9703

9.2.1 Commit and Synclevel

The Commit and Synclevel parameters are important to understand because they influence both the integrity of transactions and performance. If integrity is a requirement, a correct setting of the parameters is a requirement, but if there is no requirement for integrity (read only transactions) the same setting has a negative performance impact.

The meaning of “Commit” (CM0, CM1) can be demystified:

- ▶ **CM0: commit 0** (*commit then send*) indicates that the syncpoint commit decision is taken by IMS before message shipment occurs.
- ▶ **CM1: commit 1** (*send then commit*) means that after processing the transaction, the response is sent right away to the client and the decision to commit is returned to IMS by an external element.

The element that returns the “ACK” is dictated by the Synclevel value. Possible values are:

- ▶ *None*: IMS Connect acknowledges the response for CM1.
- ▶ *Confirm*: Used with CM0.
- ▶ *Syncpoint*: The client confirms the work, which is part of a UOW.

If you look at the different properties (persistent, shared/dedicated, clientID, and synclevel) that are mentioned in this chapter, you can think that many combinations are possible and that it is hard to find out which one to use in which conditions. In fact, the number of possible combinations is limited, and it is easy to point to the right combination that is required for a transaction.

9.2.2 Persistent Sockets

IMS TM Resource Adapter (formerly IC4J) applications running in a WebSphere Application Server can take advantage of either dedicated or shareable persistent sockets.

- ▶ A dedicated persistent socket remains dedicated to a particular user-specified clientid for commit-then-send (CM0) interactions until released by a client application's request.
- ▶ Shareable persistent sockets, on the other hand, can be shared (serially reused) by multiple applications in the WebSphere Application Server server running either send-then-commit (CM1) or commit-then-send (CM0) interactions. For the latter type of socket, IMS TM Resource Adapter generates a clientid and does not allow user-specified clientids.

For performance reasons it is important that opening and closing of sockets are not repeated but managed by a “managed connection factory”, which allows for serial reuse of existing socket connections.

9.3 High-level overview of IMS Connect

IMS Connect operates in a highly multi-tasked address space in z/OS. Its architecture was designed in layers of functionality to perform this multi-tasking. Figure 9-3 shows the core components of IMS Connect for IMS access.

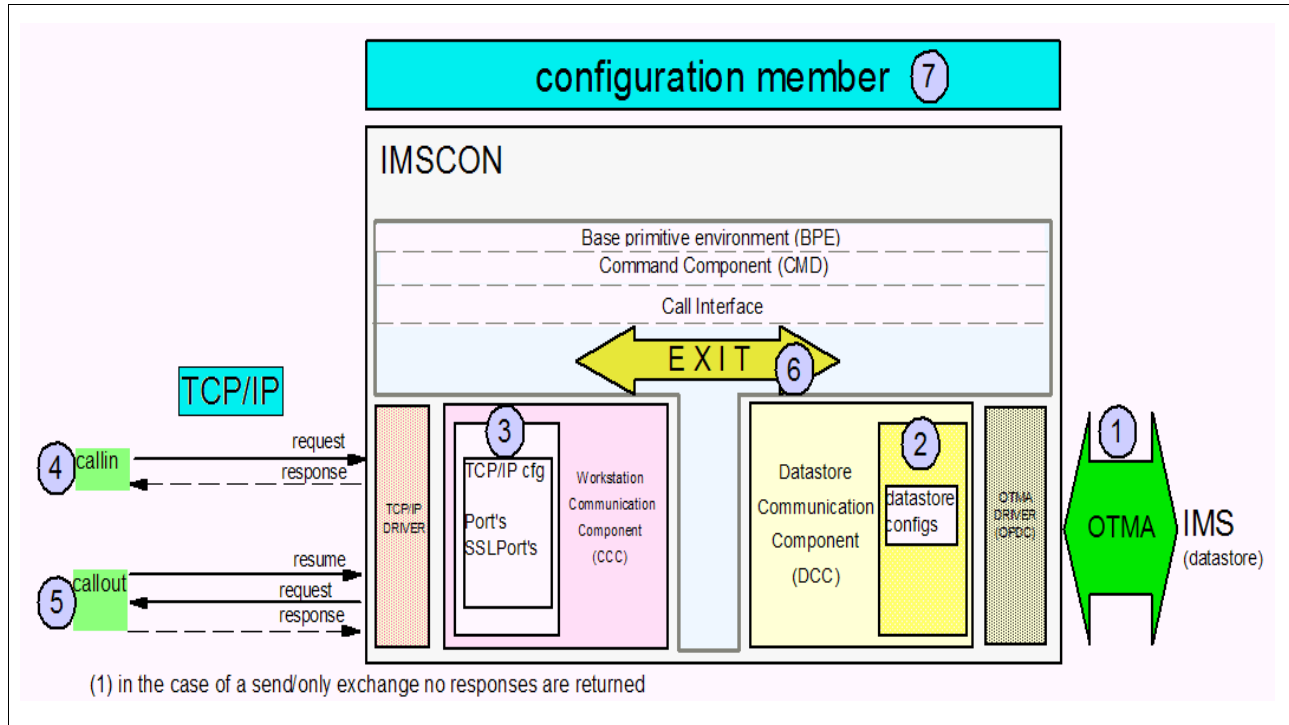


Figure 9-3 IMS Connect layout

IMS Connect performs router functions between its TCP/IP clients and IMS and IMSplex resources. Request messages that are received from TCP/IP clients, using TCP/IP connections (4) are passed to an IMS system containing OTMA (1). This IMS system, referred to as a datastore, receives communications through cross-system coupling facility (XCF) sessions through the OTMA protocol. IMS Connect receives response messages from the datastore and then passes them back to the originating TCP/IP clients (4).

The connection with consumers (callin) and producers (callout) goes over TCP/IP with an exchange of frames, which is composed of a header(s) and a payload. The content of the header and the format of the payload depends on the application, and in the case of Web Services, exchanges eventually are in XML format. The header indicates the function of the exchange. Even for the same function, the header format can be different, depending on what the participant is exchanging and what adapter technology is used.

For “callin (4)” exchanges, the client function starts with a request, which is followed or not (send/only) by a response. The exchange is controlled by several parameters or settings in the header part of the incoming message.

For “callout (5)” exchanges, although the logical client is IMS through IMS Connect, the physical client is the external partner, which uses the “RESUME TPIPE” function.

The communication component (3), provides the listening function for the TCP/IP ports. The list of active nonSSLPorts and SSLPorts used by IMS Connect is obtained from the configuration member (7).

The datastore component (2) handles the access with IMS through the OTMA protocol. The list of available IMS subsystems (datastores) is obtained from the configuration member (7).

The routing between TCP/IP to IMS, and IMS to TCP/IP is handled by IMS Connect based on header information. Exits (6) create the binding between both sides, and map headers and content. Several exits are delivered by the product and can be adapted for customer needs. These exits are:

- ▶ HWSSMPL0 and HWSSMPL1 for user-written IMS Connect client applications
- ▶ HWSJAVA0 for the IMS TM Resource Adapter
- ▶ HWSSOAP1 for the IMS SOAP Gateway
- ▶ HWSCSLO0 and HWSCSLO1 for the IMS Control Center

The selection of the correct exit is based on input from the TCP/IP side.

9.4 IMS Connect configurations

You can configure multiple IMS images on multiple z/OS systems within a single SYSplex and distribute client requests to the IMS datastores.

To configure IMS Connect:

1. The resident library in which the IMS Connect modules reside must be APF authorized.
2. Update the z/OS Program Properties Table (PPT), which allows IMS Connect to run in authorized supervisor state, key 7.
3. Enable Internet Protocol Version 6 (IPV6) for IMS Connect. Internet Protocol Version 6 (IPV6) is the next generation of the Internet Protocol that is designed to replace the current Internet Protocol Version 4 (IPV4).
4. Create an IMS Connect configuration member to hold the configuration statements that IMS Connect uses during initialization. The IMS Connect configuration member defines how IMS Connect communicates with TCP/IP, OTMA, the CSL, Open Database Manager (ODBM), and security software. The definition statements are:
 - HWS: Defines characteristics specific to an instance of IMS Connect. Specify only one IMS Connect with subparameters.
 - DATASTORE: To access IMS OTMA, specify each datastore with which the IMS Connect communicates through IMS OTMA.
 - TCPIP: Specify only one TCPIP statement to define non SSL, SSL port, and other client related connection parameters.
 - IMSPLEX: To access the IMS Operations Manager (OM), IMS Open Database Manager, or both, specify each IMSPLEX that IMS Connect communicates with through the IMS Structure Call Interface (SCI).
 - ADAPTER: Defines characteristics of adapters used to convert XML input to COBOL.

In the IMS Connect configuration member in Figure 9-4 on page 173 and detailed in Example 9-1 on page 173, IMS Connect is configured to include the ports that are defined for TCP/IP communications and the IMS OTMA group and member names for communication with IMS.

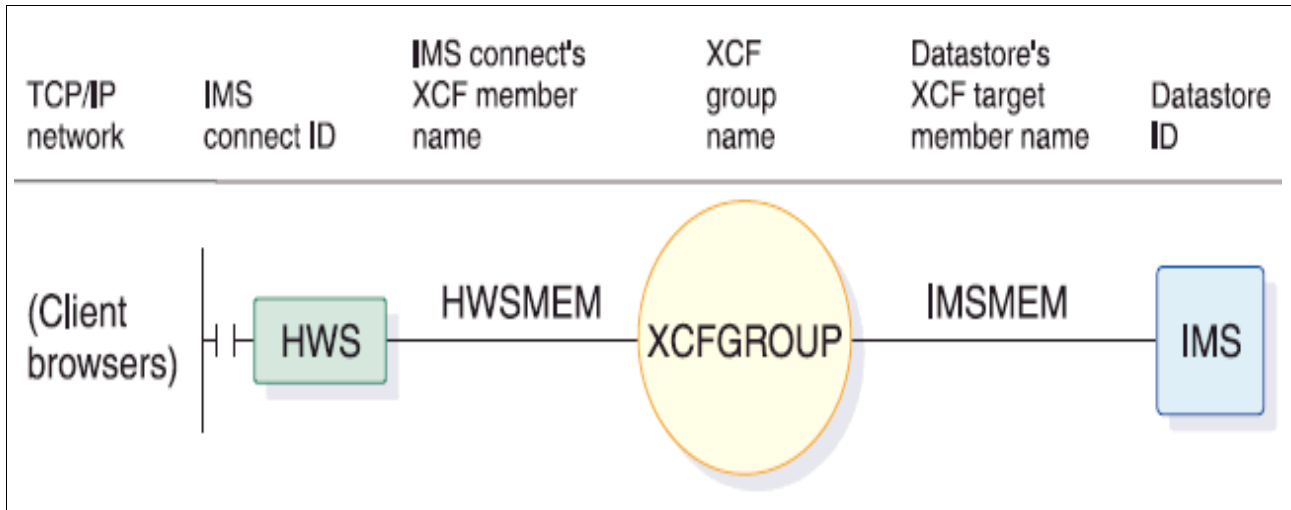


Figure 9-4 Simple configuration

- The IMS Connect ID is defined as HWS.
- The TCP/IP configuration defines the HOSTNAME as MVSTCPIP, the RACFID as RACFID, the PORTID as 9999, and the EXIT as HWSSMPL0.
- The datastore configuration defines the ID as IMS, the GROUP as XCFGROUP, the MEMBER as HWSMEM, and the TMEMBER as IMSMEM.

Example 9-1 Simple configuration example

```

*****
* IMS Connect SIMPLE EXAMPLE CONFIGURATION FILE
*****
HWS (ID=HWS,RACF=N,XIBAREA=20)
TCP/IP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999),MAXSOC=2000,TIMEOUT=8888, EXIT=(HWSSMPL0))
DATASTORE (ID=IMS,GROUP=XCFGROUP,MEMBER=HWSMEM,TEMBER=IMSMEM,DRU=HWSYDRU0)

```

In the example depicted in Figure 9-5 and detailed in Example 9-2, three IMS Connects are configured. Each IMS Connect has its own configuration member.

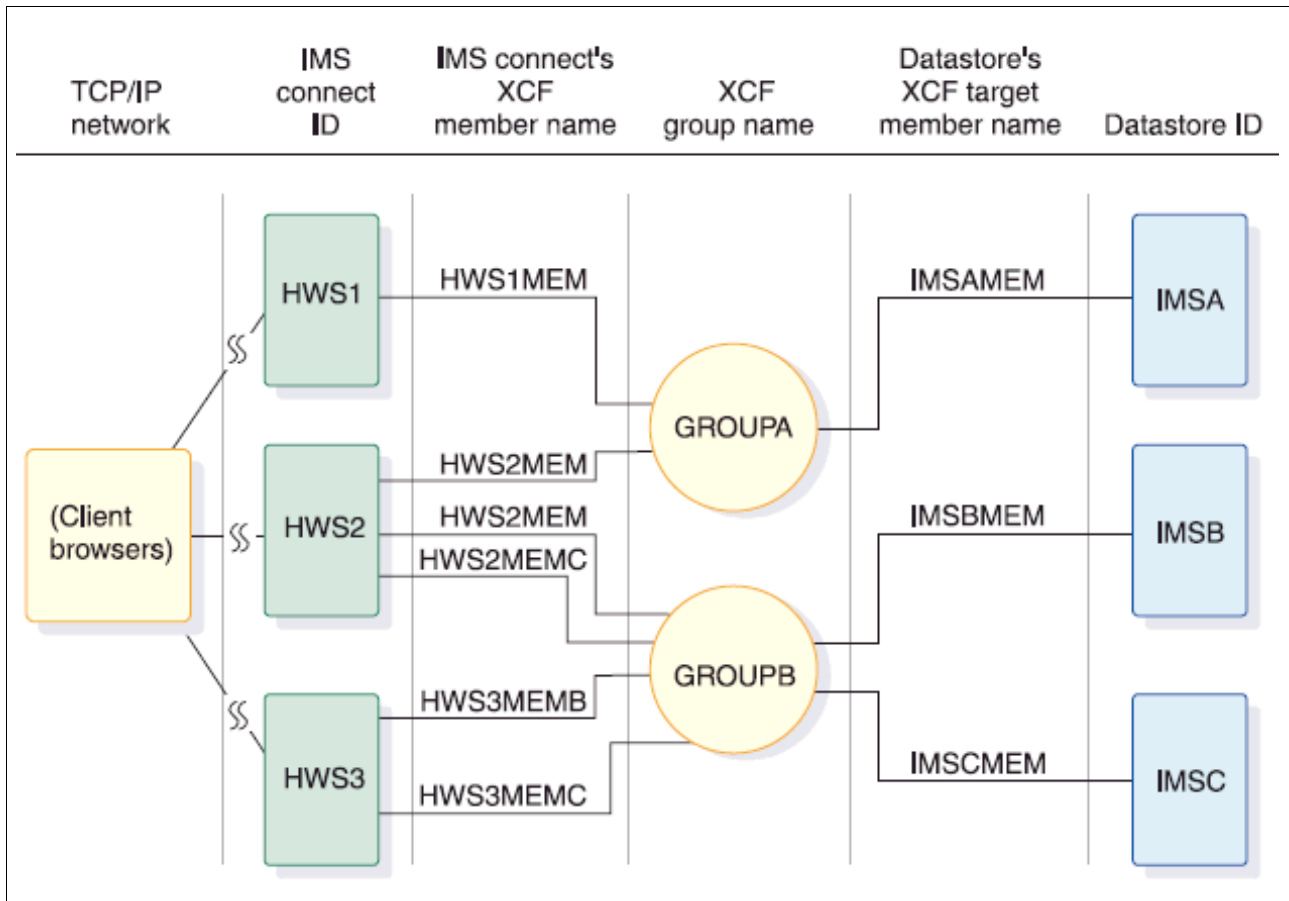


Figure 9-5 Complex IMS Connect configuration

Each IMS Connect uses a different port number for TCP/IP communications and can belong to multiple XCF groups.

When defining multiple data stores that belong to the same XCF group in a single IMS Connect configuration member, the XCF member name for that IMS Connect must be unique in each DATASTORE statement. However, if the data stores are members of different XCF groups, the XCF member names can be the same for different data stores within a single IMS Connect configuration member.

Example 9-2 Example with multiple data stores

```

*****
* HWS EXAMPLE CONFIGURATION MEMBER FOR HWS1
*****
HWS (ID=HWS1,RACF=N,XIBAREA=20)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999),MAXSOC=2000,TIMEOUT=8888,EXIT=(HWSSMPL0))
DATASTORE (ID=IMSA,GROUP=GROUPA,MEMBER=HWS1MEM,TMEMBER=IMSAMEM,DRU=HWSYDRU0)
*****
* HWS EXAMPLE CONFIGURATION MEMBER FOR HWS2
*****
HWS (ID=HWS2,RACF=N,XIBAREA=20)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9998),MAXSOC=2000,TIMEOUT=8888,EXIT=(HWSSMPL0))
DATASTORE (ID=IMSA,GROUP=GROUPA,MEMBER=HWS2MEM,TMEMBER=IMSAMEM,DRU=HWSYDRU0)

```



```

DATASTORE (ID=IMSB,GROUP=GROUPB,MEMBER=HWS2MEM,TMEMBER=IMSBMEM,DRU=HWSYDRU0)
DATASTORE (ID=IMSC,GROUP=GROUPB,MEMBER=HWS2MEMC,TMEMBER=IMSCMEM,DRU=HWSYDRU0)
*****
* HWS EXAMPLE CONFIGURATION MEMBER FOR HWS3
*****
HWS (ID=HWS3,RACF=Y,XIBAREA=20)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9997),MAXSOC=2000,TIMEOUT=8888,EXIT=(HWSSMPL0))
DATASTORE (ID=IMSB,GROUP=GROUPB,MEMBER=HWS3MEMB,TMEMBER=IMSBMEM,DRU=HWSYDRU0)
DATASTORE (ID=IMSC,GROUP=GROUPB,MEMBER=HWS3MEMC,TMEMBER=IMSCMEM,DRU=HWSYDRU0)
*****

```

Optionally, IMS Connect can be enabled for XML message conversion support. IMS Connect can convert the XML format data contained in an IMS SOAP Gateway input message into the COBOL format used by a COBOL IMS application program. The COBOL format in the corresponding output message is also converted from COBOL back to a XML format that the IMS SOAP Gateway client expects.

The IMS Connect XML-to-COBOL conversion support enables IMS to accept IMS SOAP Gateway messages in a XML format without having to create or modify IMS application programs to support XML. The definition (through the use of the ADAPTER (XML=Y) statement and specifying the HWSSOAP1 user message exit in the EXIT= parameter of the TCPIP statement) is shown in Example 9-3.

Example 9-3 IMS Connect example including XML adapter support

```

*****
* IMS Connect example of including XML adapter support
*****
HWS (ID=HWS8,RACF=Y,XIBAREA=20)
TCPIP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999,LOCAL),MAXSOC=2000,TIMEOUT=8800,
EXIT=(HWSSMPL1,HWSSOAP1))
ADAPTER (XML=Y)
*****

```

More information is in:

IMS V10 System Definition Guide, GC18-9998 (Chapter 9 Integrated IMS Connect definition and tailoring)

9.5 What is new in IMS Connect Version 10

IMS Version 9 already provides an integrated IMS Connect function, which is robust and functionally rich. More improvements were introduced with IMS Version 10. For more details refer to *IMS Version 10 Implementation Guide A Technical Overview*, SG24-7526.

The key features are:

- ▶ OTMA and IMS Connect added support for RESUME TPIPE with an Alternate Clientid (TPIPE).
- ▶ IMS Connect is enhanced with a “send-only with acknowledgement” protocol. When the send-only with acknowledgement protocol option is specified, the client application receives an ACK response message from OTMA for each input message successfully enqueued by IMS. All other output that is generated by the send only transaction is sent to the asynchronous hold queue.

- ▶ IMS Connect is enhanced with another option of “send-only with serial delivery” using the DELIVERMSG=ORDERED option of the XCF IXCMMSGO macro. When the send-only with serial delivery protocol option is specified, IMS Connect ensures that the order in which it submits send-only transactions to OTMA is in fact the order in which IMS receives the transactions.
- ▶ The IMS Connect XML Adapter support for COBOL and PL/I, together with the IMS SOAP Gateway and WebSphere Developer for System z (WDz) or Rational Developer for System z (RDz), enables the reuse of IMS applications as Web Services. The COBOL support was also provided through the IMS SOAP Gateway V9.2 and IMS V9 service process.
- ▶ The IMS Connect performance enhancement introduces the new NODELAY configuration statement parameter. Using the NODELAY option can enhance IMS Connect performance and increase the throughput because it forces a socket to send the data in its buffer without having to wait for an ACK response from the client’s TCP/IP. The NODELAY parameter is used only when sending packets.
- ▶ The IMS Connect security enhancements introduce the following new functions into IMS Version 10:
 - IMS Version 10 users of the user exit routines HWSSMPL0, HWSSMPL1, and HWSJAVA0 can submit input messages to change the passwords that are used by IMS Connect client application programs. The routines check for a leading keyword of ‘HWSPWCH’ to determine whether it is a request to change the password. This ‘HWSPWCH’ string can be viewed as a transaction code, but new logic in HWSPWCH0 is called to process the special request.
 - Users that configure RACF to accept mixed-case passwords can configure IMS Connect to allow mixed-case passwords also.
 - Users can define in the configuration member of IMS Connect an aging value for the accessor environment element (ACEE) that OTMA uses for security checking of the clients associated with the IMS Connect. The ACEE control element represents the credentials of a user linked to a z/OS task.
- ▶ IMS SOA Composite Business Application support is a new capability for IMS TM Resource Adapter clients that invokes IMS conversational transactions. Although the IMS TM Resource Adapter supported interactions with IMS conversational transactions, IMS Version 10 enhances this capability by allowing the iterations between a conversation to span multiple shareable sockets.

9.6 What is new in IMS Connect Version 11

IMS Version 11 provides enhancements to IMS Connect for both IMS DB and IMS TM environments.

9.6.1 Integrated IMS Connect enhancements for IMS DB

The enhancements to IMS Connect for IMS DB include a new Distributed Relational Database Architecture™ (DRDA) compliant application programming interface (API) and the ability to communicate with the new IMS Open Database Manager address space.

With IMS Version 11, IMS Connect is the TCP/IP path into IMS DB and IMS TM. IMS Connect clients access IMS DB through a new API, one that adheres to the open DRDA standard specifications and supports Distributed Data Management (DDM) architecture commands.

As part of the new Open Database enhancements, IMS Version 11 provides new Java drivers, called the IMS Universal drivers, that you can use to access your IMS data. IMS Connect supports the IMS Universal drivers with the DRDA API. Independent software vendors can also use any of the IMS Universal drivers to build packages that access IMS data.

For more information about Open Database and the IMS Universal drivers, refer to Chapter 15, “IMS Open Database and Universal Drivers” on page 321.

9.6.2 Integrated IMS Connect enhancements for IMS TM

The IMS Connect enhancements for IMS TM provide better connectivity, serviceability, security, and usability. The IMS Connect enhancements for IMS TM provide:

- ▶ IMS Connect configuration member HWSCFGx enhancements

New parameters were added to IMS Connect configuration statements of HWS, TCPIP, and DATASTORE for:

- CM0 ACK time-out support

The HWS statement provides a new CMOATOQ value that defines a global override time-out queue name for all the datastores (IMS systems) accessed by an IMS Connect instance. The IMS Connect CMOATOQ value overrides the default name in OTMA

- Super member support at a datastore level

The DATASTORE statement was enhanced to support a new parameter for super member support. The new SMEMBER parameter on individual DATASTORE statements overrides the default HWS value. By providing the super member support at a datastore level, IMS Connect is now able to support multiple super member names.

- Inactive socket detection at a PORT level

IMS Connect does not support multiple SSL ports. Previous releases of IMS Connect allowed the specification of more than one SSL port but did not actually support running more than one concurrently. Attempting to actually running multiple SSL ports with multiple sockets with any level of concurrent processing caused a failure in the Language Environment.

IMS 11 restricts users from specifying more than one port in the configuration member. If more than one SSL port is specified on the SSLPORT= parameter, IMS Connect displays error messages to the system console and then abend.

- Warning messages and early detection of maximum sockets

IMS Connect supports between 50 and 65,535 sockets. The maximum number of sockets that an IMS Connect instance supports is specified in the MAXSOC= parameter. When the number of sockets reaches the MAXSOC limit, any new connections are refused and message HWSS0771W is issued. IMS Version 11 introduces the WARNSOC and WARNINC parameters as ways to provide early detection of this potential problem. WARNSOC supports a decimal value between 50% and 99% as a warning level.

- ▶ Enhanced IMS Connect commands

Although the IMS Connect VIEWHWS command (using the WTOR interface) and the QUERY MEMBER command (using the z/OS modify command interface) provide a complete picture of the status of the IMS Connect resources, the resulting display can generate many lines of output that can flood the console, particularly if many active

sockets were established. The addition of the SUMMARY option for both these commands allow PORT status and client totals to be displayed, but bypass the individual listing of each individual socket for the PORTs.

▶ Port Input/Output Edit Exit

The new Port Input/Output Edit Exit provides an exit interface for remote clients that cannot conform to the IMS Connect message format, e.g., pass in a full IRM header, but still need the functionality of the support.

▶ HWSTECL0 event record enhancements

IMS Connect facilitates event recording by passing event data to the load module, HWSTECL0. This module stores all trace and event notifications through a recording routine and can be used by any event recording function. IMS 11 produces four new event records for HWSTECL0 that report the activity through the Port I/O Exit.

▶ HWSEXPRL parameter list expansion

IMS Version 11 expands the existing HWSEXPRL macro that is used by IMS Connect user message exit routines. The reasons for this area:

- Improves readability and serviceability of the parameter list.
- Reorganizes the current fields for better placement and eliminates discrepancies caused by lack of space.

▶ User message exit cleanup

HWSIMSO0 and HWSIMSO1 message user exits are no longer shipped with IMS Connect. HWSSMPL0 and HWSSMPL1 user exits provide enhanced functionality and are delivered as source code replacements, which also reduces the number of user message exits to maintain.

▶ Additional information in the HWSP1410W 'failed to release storage' message

Existing messages already include the error return code, type of storage, and the module that encountered the error. The address of the storage in error is now provided.

▶ Cancel Client ID support

The connection between a remote client application and IMS Connect sometimes gets disrupted due to TCP/IP failures, or processing failures on the client side. Because the original client status might still be active in IMS Connect, a request to reestablish the connection to the same port using the same clientid can result in a duplicate client condition and message HWSS0742W.

With this new function, the client application can establish a new connection with a request to "Cancel Client ID" which causes IMS Connect to automatically discard and cleanup any previously active session for that client ID. All CM0 or CM1 transactions running on either a persistent or transaction socket are supported.

▶ TCP/IP auto reconnect

If the TCP/IP network experiences a disruption during IMS Connect processing, operators must issue the OPENPORT command when the network is once again available. With the TCP/IP Auto Reconnect support in IMS Version 11, the same situation no longer requires operator intervention because IMS Connect continues to listen on the socket associated with each active port and automatically attempts to reestablish a connection as appropriate.

- ▶ **IMS Connect performance enhancement**
A new hashing mechanism for client ids enhances IMS Connect performance by reducing CPU overhead and increasing throughput. This support is also available to IMS Version 10 through APAR PK57574 through PTF UK42318.
- ▶ **New recorder trace and BPE support**
IMS 11 introduces a new tracing capability for IMS Connect that takes advantage of the BPE external trace data set support. This enhancement addresses the problems associated with the limitations of the existing capability which is based on a dataset defined to the IMS Connect startup JCL.

IMS Version 11 has many important features that are available for IMS Connect users.

9.7 IMS as a service provider

With the enhancements provided by IMS Versions 10, 11, and the set of tools in the IMS SOA Integration Suite, IMS is now in the position of being both a service provider and requestor in the SOA world. In this section, we list the possible service consumers that are connected through IMS Connect to IMS.

Consumers of an IMS service

We distinguish four kinds of service consumers processing through IMS Connect. Figure 9-6 presents these interfaces:

- ▶ IMS Control Center (1)
- ▶ RYO (Roll Your Own) client (2)
- ▶ IMS TM Resource Adapter (TMRA) (3)
- ▶ IMS SOAP Gateway (4) and any web service clients (5) interfacing with it

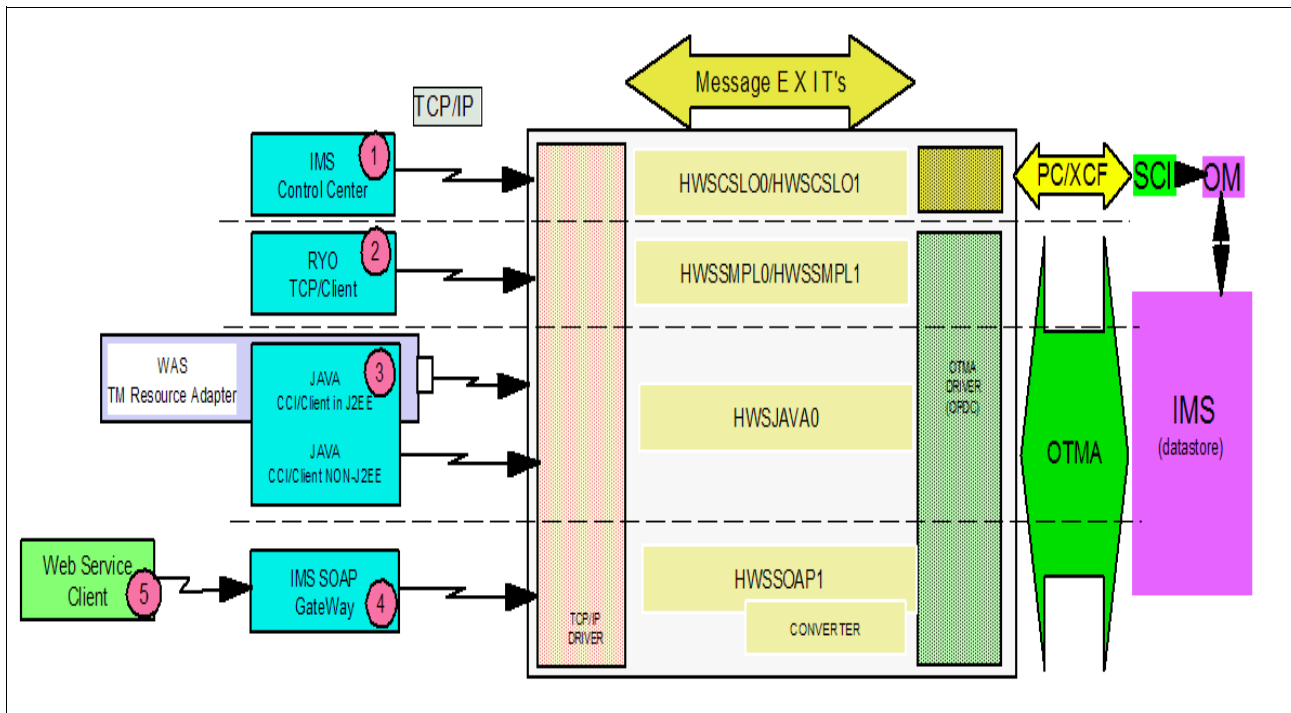


Figure 9-6 IMS as service provider

Next, we describe each of these IMS service consumers in more detail.

9.7.1 IMS Control Center

You can manage your IMS systems in a graphical interface from a workstation using the IBM DB2 V9.1 for Linux, UNIX, and Windows Control Center. Using the Control Center for IMS provides a single point of control that simplifies system management tasks in an IMSplex environment. Administrators can work with a consolidated system view and manage all their IMS systems from a single station.

You can issue and view IMS type-1 and type-2 commands from the IMS Control Center. There are online wizards that help you build and issue commands. The Control Center command output is similar to the output displayed from the TSO ISPF based SPOC. The same information is provided but formatted in a windows-based graphical interface.

The exit routines HWSCSLO0 and HWSCSLO1 are delivered as Object Code Only (OCO) with IMS Connect to allow IMS Connect and the user message exit for the Control Center to be synchronized without requiring simultaneous upgrades of other products when message exit functions change. If your installation activates the Control Center to communicate with OM, you must include the HWSCSLO0 and HWSCSLO1 exit names in the EXIT= parameter of the TCPIP statement.

The HWSCSLO0 and HWSCSLO1 message exits are designed to be used only by the IMS Control Center client and cannot be used by any client that sends messages to a data store.

The default COMMIT mode is set to 1, and the SYNC level is set to NONE. You can override these values.

The HWSCSLO0 exit performs translations from EBCDIC to ASCII and removes the internal headers for messages being transmitted to the client. Exit HWSCSLO1 does not perform any translation on the output data, but it does remove the internal headers for messages being transmitted to the client. Figure 9-7 on page 181 displays the members of the IMSplex PLEX1 using the IMS Control Center.

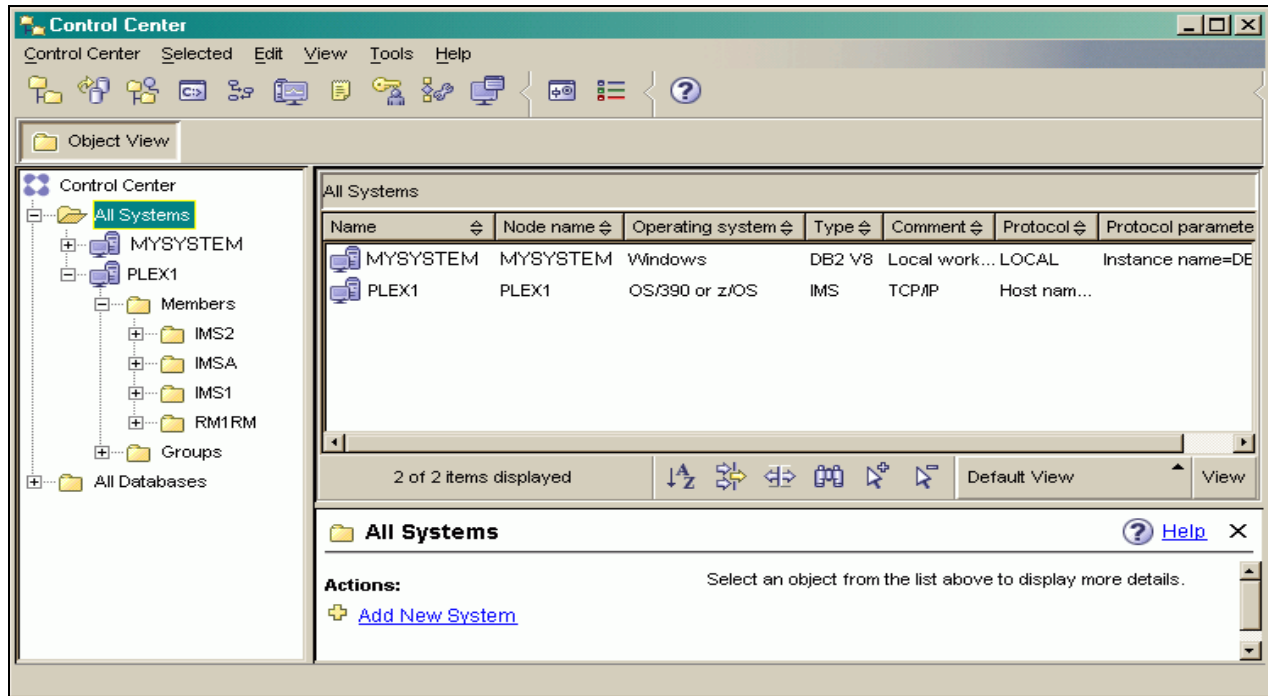


Figure 9-7 IMS Control Center

9.7.2 Roll Your Own client program

You can send transactions and commands to IMS using IMS Connect, but without using the TM Resource Adapter. We call these Roll Your Own (RYO) clients. Programs can be written in several languages, such as C and Java. The only requirement is to have TCP/IP access.

The structure of an IMS Connect message is simple, it is composed by a message header followed by the message data. The header structures you use must correspond to what the IMS Connect user message exit expects to find. Two sample exits are provided with IMS Connect, HWSSMPL0 and HWSSMPL1. If your installation uses a modified exit, adapt the code accordingly with those changes.

The header structures are described in the following references, which also contain C language examples:

IIIMS Connectivity in an On Demand Environment: A Practical Guide to IMS Connectivity, SG24-6794

SC18-9703 IMS V10 Communications and Connections Guide, SC18-9703

To achieve the right interaction between the client and IMS Connect to and from IMS, four types of special structures were architected:

- ▶ **IMS Request Message (IRM)**

These are the messages sent from the client to IMS Connect.

- ▶ **Request Status Message (RSM™)**

This is the message returned by IMS Connect to the client when an error occurred. The RSM contains a return code and a reason code identifying the type of status.

- ▶ Complete Status Message (CSM)
IMS Connect sends this message as the last structure of an output message if the input message was processed successfully.
- ▶ Request MOD Message (RMM)
IMS Connect returns the RMM as the first structure of an output message if the MFS MOD name is requested and there is output data present.

The IMS Connect input message

The input message is actually what the client application sends to IMS Connect. The structure of the input message is always the same. See Example 9-4 for its layout.

Example 9-4 Input message

```
LLLL IRM LLZZdata1 LLZZdata2 ... 0004 0000
```

Valid message types that are indicated in the header (IRM_F4 byte) are:

- ▶ 'A' ACK (IRM_F4_ACK)
An ACK response to output received from IMS Connect. ACK is used by a client to indicate the acceptance of an output message only when the original input message from the client specifies a SYNC level of CONFIRM.
- ▶ 'C' Cancel IRM timer (IRM_F4_CANTIMER)
A request to cancel the IRM timer for another connection on which the client, using the same client ID, is waiting for output data.
- ▶ 'D' Deallocate (IRM_F4_DEALLOC)
A request to deallocate the conversation.
- ▶ 'K' Send only requires ACK (IRM_F4_SNDONLYA)
A send-only transaction message that requires an ACK response from IMS Connect.
- ▶ 'N' NAK (IRM_F4_NACK)
A NAK response to output received from IMS Connect. NAK is used by a client to indicate the rejection of an output message only when the original input message from the client specifies a SYNC level of CONFIRM.
- ▶ 'R' RESUME TPIPE (IRM_F4_RESUMET)
A RESUME TPIPE call for asynchronous output data from IMS. A RESUME TPIPE call must execute on a transaction or persistent socket using CM0.
- ▶ 'S' Send only (IRM_F4_SENDOONLY)
A send only transaction message that executes a send-only interaction for a non-response mode, non-conversational transaction.

The IMS Connect output message

The output message is what IMS Connect sends back to the client application.

The structure of the output message depends on whether we requested a MOD name to be present (setting IRM_F1 to X'80') and the outcome (success or failure) of the interaction. We also must take into account that if we use the HWSSMPL1 exit, IMS Connect places a four-byte field with the total length, which includes these four bytes before the output message. Example 9-5 on page 183 presents some of these output message formats.

Example 9-5 Output message

```
[LLLL] RMM LLZZdata1 LLZZdata2 ... CSM  
[LLLL] LLZZdata1 LLZZdata2 ... CSM  
[LLLL] RSM
```

Those three formats correspond to:

- ▶ A successful interaction, with MOD NAME requested.
- ▶ A successful interaction, without MOD NAME requested.
- ▶ A non-successful interaction.

9.7.3 IMS TM Resource Adapter client program

Although access to IMS transactions using IMS Connect can be controlled by non-Java clients, it is most commonly performed by Java clients on a local host or a distributed platform. Those Java clients are using a Java connector based on the “Client Connector Interface (CCI)” architecture. We distinguish two types of client/connector code:

- ▶ Java connector code, called by a Java EE artifact (Servlet, or EJB) in the WebSphere Application Server. The socket connection established for the interaction with IMS Connect/IMS can be delivered by a Connection Factory, defined in WebSphere Application Server.

The program requests a connection at the factory, and the manner it is obtained (new or reused) is the responsibility of the factory. WebSphere Application Server use of reuse and socket pooling avoids multiple physical opening and closing of sockets.

- ▶ Stand alone clients must build a runtime Connection Factory.

The connector code is, for the most part, a result of the generation by wizards in an Eclipse based Rational Application Developer workbench.

The current version for Rational Application Developer is Version 7. The generated Connector code can be used in several environments and by different types of clients. The J2C client code stub, obtained by the Rational Application Developer wizard, can be extended in several ways through options in the Rational Application Developer, as indicated in Figure 9-8 on page 184. It can be:

- ▶ Prepend/called by a session EJB (WebSphere Application Server)
- ▶ Instantiated by a Java Server Page (JSP) (WebSphere Application Server)
- ▶ The backend of a Web Services service.

The combined implementation must be deployed as a Java EE application in a WebSphere Application Server.

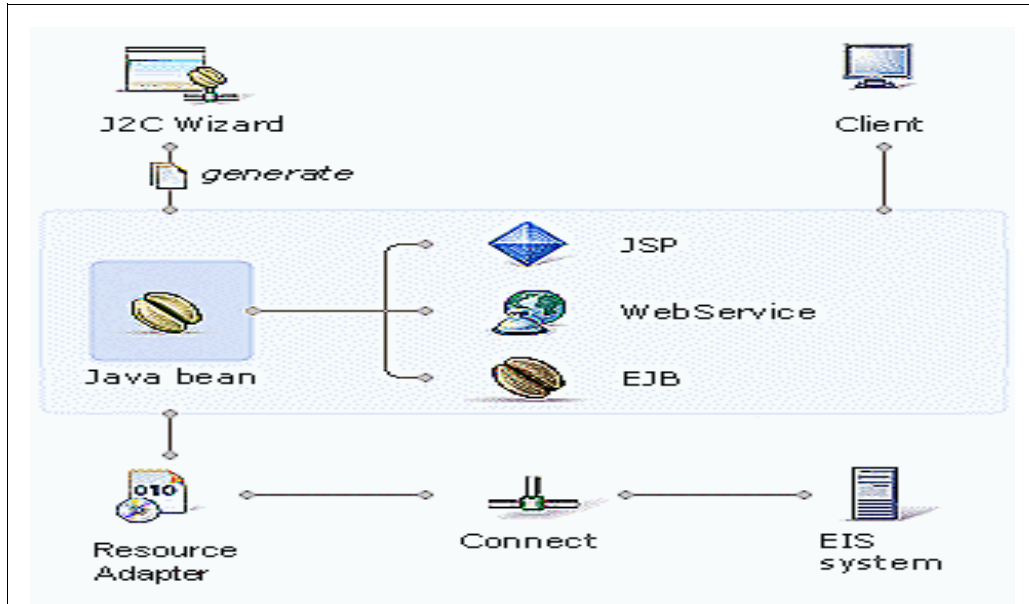


Figure 9-8 J2C wizard

With the WebSphere Integration Developer (WID) extension of Rational Application Developer, you can build an importable Service Component Architecture (SCA) component for an invocable IMS transaction. This component can then be imported by a Mediation Module (for WebSphere Enterprise Service Bus (WESB)) or a Module (for WebSphere Process Server(WPS)).

As you can see in Figure 9-9, to import a J2C based service in a module, an “EIS import” binding is available. To prepare such an import, a WID extension of RAD is required. You can use other import bindings through Web Services and Enterprise Session Beans.

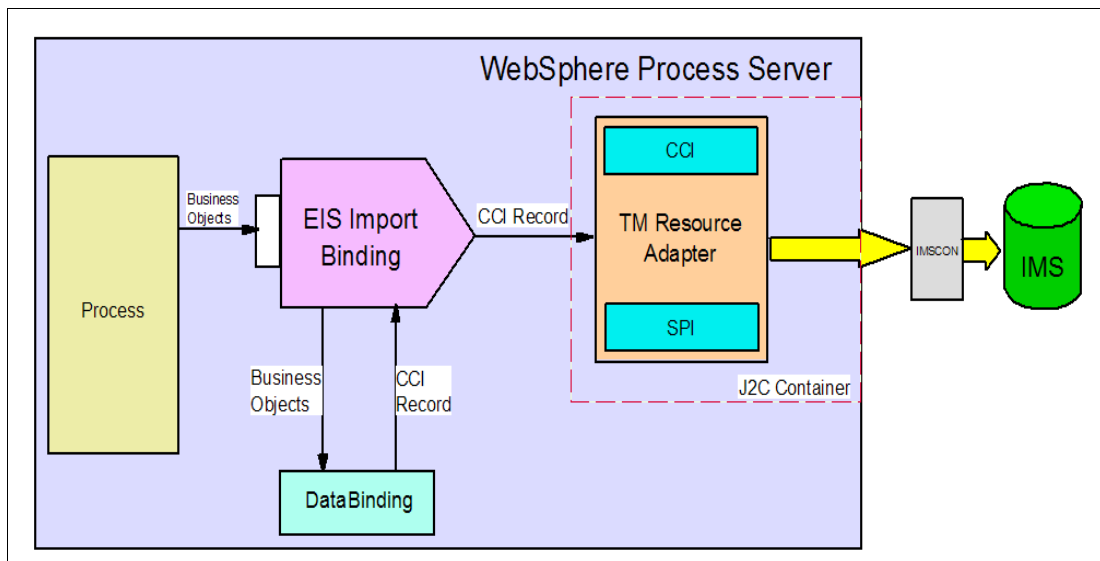


Figure 9-9 EIS import binding of an IMS connector

9.7.4 Soap Gateway

The Soap Gateway role is to make an IMS transaction look like a Web Service, and to be a SOAP service provider.

Messages used in Web Services have a particular format based on XML, called a SOAP frame. Although it is possible to handle those messages directly in the message processing program (MPP), most of the time people prefer to use existing transactions without change. In this case, it is required to have a function that converts flat message layouts into the SOAP format, and vice versa. If this is not done by a process external to IMS, then MPPs must deal with the XML layout.

Note: Keep in mind that although IMS messages can contain XML delimiters, if they handle the XML aspects, some required fields are an exception to this statement, which is the case for “ll, zz transaction code”.

The gateway requires several ingredients:

- ▶ **Connection bundle:** Describes the connection with IMS Connect and security. Define one connection bundle for each IMS system that IMS SOAP Gateway is communicating with to access an IMS application as a Web service. All the Web Services can share the same connection bundle if they run in the same IMS instance.
- ▶ **Correlator file:** Specifies the transaction, runtime, and correlation properties that IMS SOAP Gateway needs to match incoming requests to the appropriate back-end IMS application. Every Web service must have its own correlator file with the name of the connection bundle defined.
- ▶ **XML converter program:** Called by the HWSSOAP1 adapter exit for the XML conversions.

All of these elements can be built with Rational Application Developer/z wizards. For more, “how to” information about this subject, refer to Chapter 10, “The IMS SOAP Gateway” on page 203.

9.8 IMS as a service consumer

IMS Callout refers to the ability of an IMS application to act as a client to invoke a service that is external to the IMS environment (for example, an application that is running in the distributed platform). Depending on the service we probably have some questions to answer:

- ▶ What is the protocol to be used, how can we reach the other service?
- ▶ Do we need security settings to reach the service?
- ▶ What kind of service is it? Is it update or query?

Also we must decide on the use of certain options:

- ▶ Synchronous, or asynchronous call out flow
- ▶ Send_only (no response), or Send_Receive

Callout partners for IMS through IMS Connect

Figure 9-10 on page 186 presents the facilities that are available to create callout solutions:

- ▶ **IMS TM Resource Adapter (asynchronous callout mode):**
 - Through a StateLess Session Bean (SLSB in WebSphere Application Server)
 - Through a Message Driven Bean (MDB in WebSphere Application Server)

- ▶ IMS TM Resource Adapter (synchronous callout mode):
 - Through a StateLess Session Bean (SLSB in WebSphere Application Server)
 - Through a Message Driven Bean (MDB in WebSphere Application Server)
- ▶ IMS SOAP Gateway (asynchronous callout mode)
- ▶ IMS SOAP Gateway (synchronous callout mode)

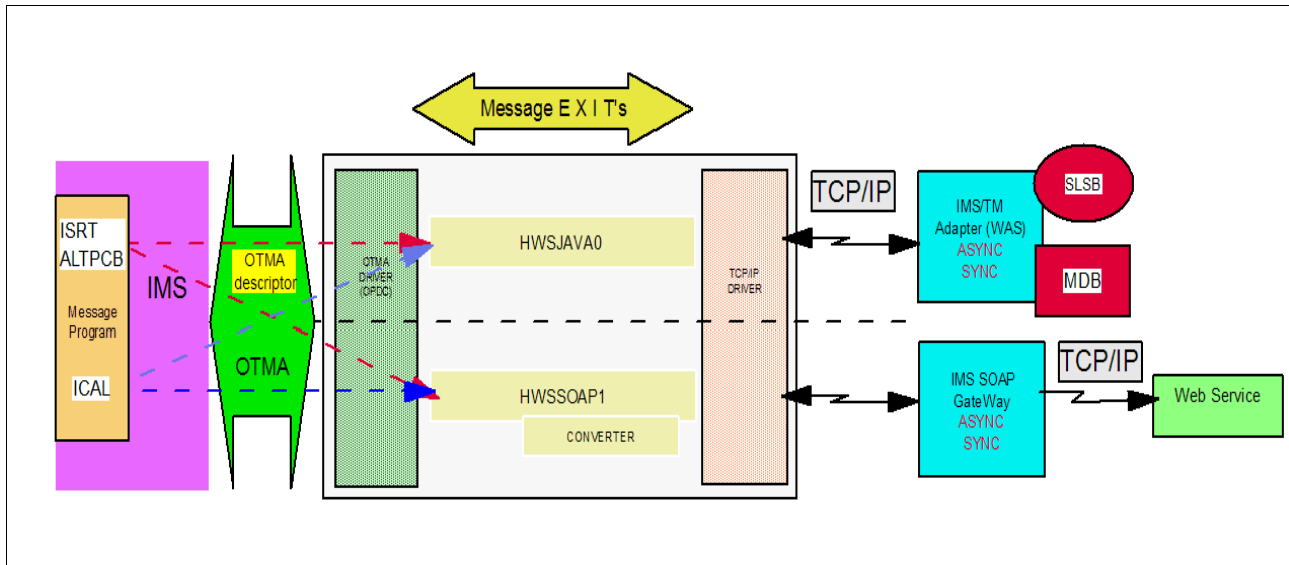


Figure 9-10 IMS as a service consumer

Before we discuss the different consumer possibilities, we want to point out an aspect of IMS Connect communication. Although in these scenarios IMS Connect is the requester, from a TCP/IP point of view it is always a listener. The first call from the participant (SOAP Gateway, or TM Resource Adapter) is always a *Resume TPIPE* call that can be interpreted as an invitation to IMS Connect for sending.

9.8.1 IMS TM Resource Adapter asynchronous callout mode

IMS applications today can issue the ISRT ALTPCB call to direct output messages to another IOPCB. This line item leverages the existing ISRT ALTPCB call to allow IMS applications to "send" callout requests to an asynchronous hold queue (also referred as a TPIPE). At this point, the callout request is retrieved by IMS TM Resource Adapter

In asynchronous callout mode, the IMS application calls out to an external application and terminates. The output (if any) is sent back to another IMS application instance to continue processing. The advantage of using asynchronous callout is it does not hold up the region, but it does require IMS applications to be designed in a way that they are able to handle the output from the external application in a different IMS application instance.

9.8.2 The IMS TM Resource Adapter synchronous callout mode

Using support provided through the maintenance stream in IMS Version 10 and distributed with IMS Version 11, IMS applications can issue the "DL/I ICAL call to callout synchronously.

The advantage of using synchronous callout is that the response can return to the waiting IMS application as a message rather than having to spawn a subsequent IMS transaction.

But it requires the IMS dependent region to be held up in a wait state while waiting for the output from the external application to return.

For more information on both the asynchronous and synchronous callout support delivered with IMS TM Resource Adapter, refer to Chapter 11, "The IMS TM Resource Adapter" on page 225.

9.8.3 IMS SOAP Gateway callouts

To callout to an external Web service, an application developer can choose to build a new IMS application specifically created for calling out, or modify an existing IMS application to make the callout request message. These two scenarios can be described as "top-down" and "meet-in-the-middle" models.

- ▶ In the "top-down" model, a brand new IMS application is created and developed to make the callout request message. In this case, a new data structure is created (or generated by the tooling), based on the input data structure of the external Web service.
- ▶ In the "meet-in-the-middle" model, the user modifies their existing IMS application to make the callout request message. In this case, there is an existing structure in the IMS application that is used to make the callout request message. Because the data structure of the external Web service is already predefined, it is unlikely that the data structure of the Web service is exactly the same as the data structure used by the IMS callout application. Therefore, some data mapping needs to take place between the data structure of the IMS application issuing the callout, and the input data structure of the external Web service.

IMS SOAP Gateway enables IMS applications to inter operate outside of the IMS environment through SOAP to request services that are independent of platform, environment, application language, or programming model.

IMS SOAP Gateway is compliant with industry standards for Web Services, which includes SOAP/HTTP 1.1 and Web Services Description Language (WSDL) 1.1. This compliance enables your IMS assets to interpret openly with various types of applications.

The questions that are associated with the outbound call to Web Services are:

- ▶ Is the call asynchronous (no wait for response) or synchronous (how long must we wait)?
- ▶ Who is responsible for building the SOAP messages? This can be the IMS program, or we can leave it up to IMS Connect and the adapter/converter function.

XML data transformation is optional for both the callout request and response. If XML data transformation is not used, the IMS application must send the callout request message in a XML format that can be processed by IMS SOAP Gateway and the target Web service appropriately.

If the user specifies IMS Connect to perform data transformation using the XML Adapter conversion function, the IMS application's data is converted to a XML message designated for a particular Web service operation. The IMS Connect XML conversion function requires a XML converter for each Web Service operation the IMS application is calling out to. The XML converter can be generated from Rational Developer for System z using the Web Service Definition Language (WSDL) file of the Web Service.

IMS SOAP Gateway asynchronous callout mode

IMS applications can issue the ISRT ALTPCB call to direct output messages to another destination. This capability is used to allow IMS applications to send a callout request to an

asynchronous hold queue (also referred as a TPIPE) using ISRT ALTPCB. IMS SOAP Gateway implements the IMS Resume TPIPE protocol and manages the Resume TPIPE loop inside the IMS SOAP Gateway server to continually retrieve asynchronous messages that were inserted from alternate PCB destinations by IMS applications.

After the callout request message is received, IMS SOAP Gateway processes the request, finds the Web service identifier in the callout request message and matches it with the deployed Web Services to locate and invoke the desired Web Service. The invoked Web Service either performs the task and ends or sends the callout response message back to IMS asynchronously to start a new IMS transaction.

For more details, refer to Chapter 10, “The IMS SOAP Gateway” on page 203.

IMS SOAP Gateway synchronous callout mode

An IMS application program can issue an ICAL SENDRECV call to send out a synchronous callout request. The call specifies an OTMA descriptor name which defines the destination for the call.

To support IMS Synchronous Callout to Web Services, IMS SOAP Gateway implements the enhanced IMS Resume TPIPE protocol and manages the Resume TPIPE loop inside the IMS SOAP Gateway server to continually retrieve synchronous callout messages that were inserted by the IMS application. It is recommended to not have both asynchronous and synchronous callout messages on the same TPIPE unless there is a strong reason.

The data transformation for synchronous callout is same as the asynchronous callout function. The only difference is the IMS Connect XML adapter function can handle messages larger than 32K. It relieves the 32K segmentation limitation for synchronous callout messages.

After the synchronous callout request message is received, IMS SOAP Gateway processes the request, finds the Web service identifier in the callout request message, and matches it with the deployed Web Services to locate and invoke the desired Web Service. The invoked Web Service, after processing, sends the callout response message back to the IMS application that is waiting synchronously for a response.

For more details, refer to Chapter 10, “The IMS SOAP Gateway” on page 203.

9.9 IMS Connect Extensions

This section applies to Version 2 Release 1 of IBM IMS Connect Extensions for z/OS (product number 5655-S56). IMS Connect Extensions extends IMS Connect by providing:

- ▶ Monitoring and recording of IMS Connect activity
IMS Connect Extensions provides a detailed audit of activity by providing you with the information you need to analyze performance, throughput, resource availability, and security.
- ▶ Single point of control for multiple IMS Connect systems:
This tool supplies centralized management and control of all your IMS Connect systems from an ISPF dialog.

- ▶ Enhanced transaction management

Dynamic management of TCP/IP transactions is available to allow you to define rules to automatically distribute workloads and reroute messages when network failures occur.
- ▶ Improved client services

IMS Connect Extensions provides for enhanced information in error messages, a password change facility, and extended message translation.

IMS Connect Extensions runs in the same address space as IMS Connect, as seen in Figure 9-11. Detailed information can be found in the manual:

IBM IMS Connect Extensions for z/OS V2.1 User's Guide, SC19-1163

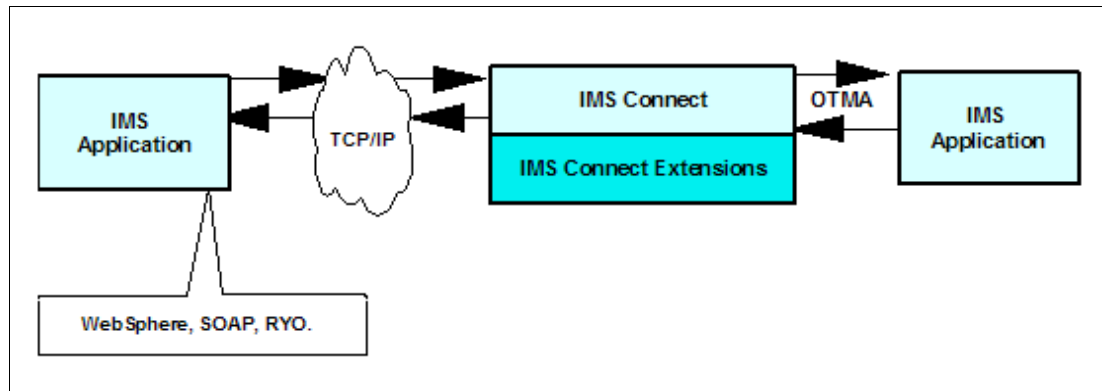


Figure 9-11 Position of IMS Connect Extensions assisting IMS Connect

9.9.1 IMS Connect Extensions components

IMS Connect Extensions consists of several components. Some components run in the IMS Connect address space and are initiated by IMS Connect when it starts.

The main components of IMS Connect Extensions are:

- ▶ ISPF dialog client

The ISPF dialog client stores configuration settings in a VSAM repository. The ISPF client connects using TCP/IP to one or more IMS Connect Extensions consoles, providing centralized monitoring and control of IMS Connect systems across your enterprise.
- ▶ Console

This is an agent running in the IMS Connect address space that listens on a TCP/IP port for connections from IMS Connect Extensions ISPF dialog clients. It allows clients to access information about an IMS Connect system and issue commands to that system.
- ▶ Repository

This is a VSAM Key Sequenced Data Set (KSDS) that contains configuration data for IMS Connect Extensions. You can manage the repository using the ISPF dialog client or the batch definition utilities.

Figure 9-12 on page 190 provides more detail on IMS Connect Extensions components.

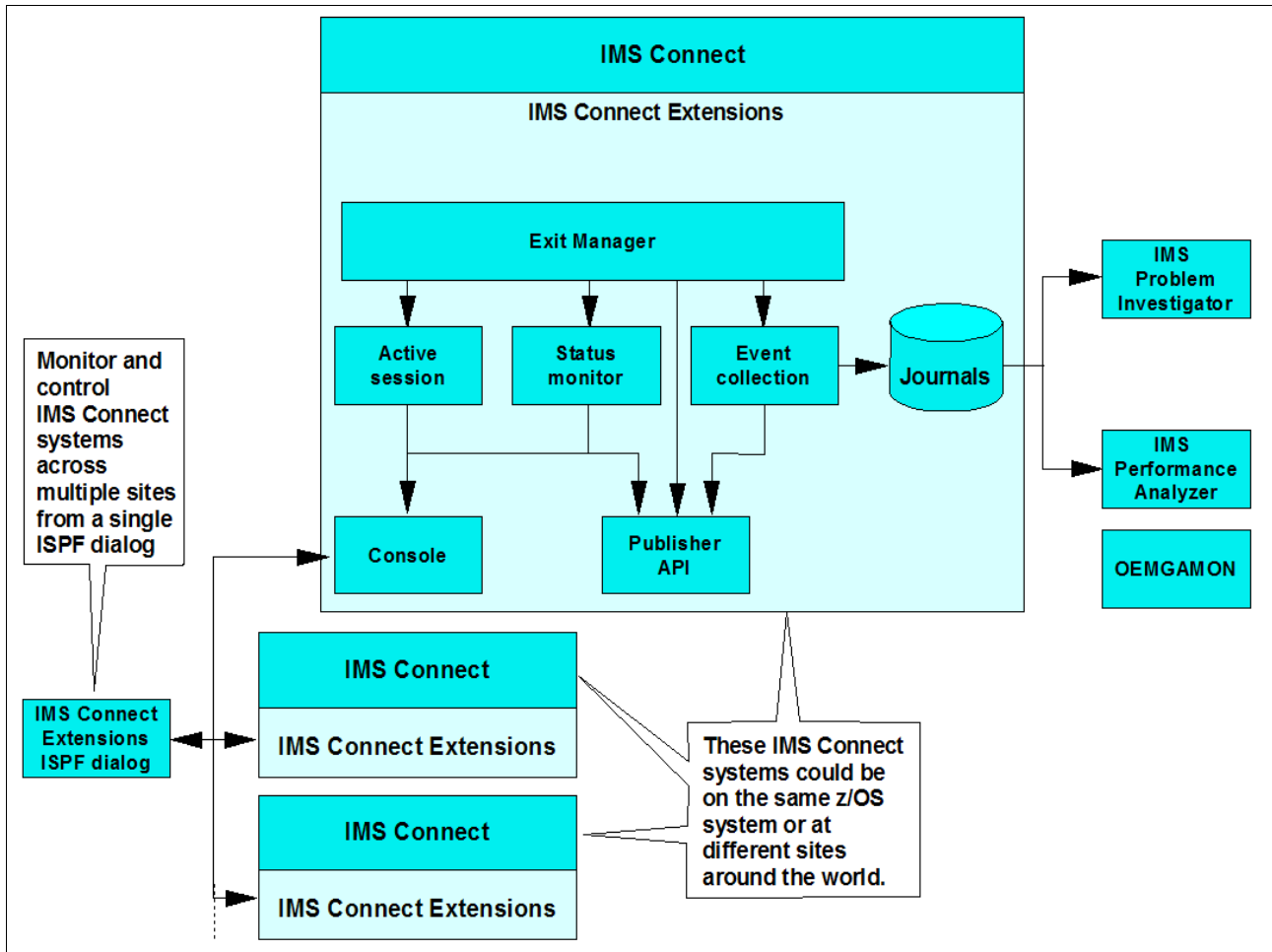


Figure 9-12 IMS Connect Extensions components

► Exit manager

When IMS Connect starts, it initializes the IMS Connect Extensions user exit manager. The user exit manager loads all user exits on behalf of IMS Connect and dynamically configures event collection, real-time monitoring, message translation, and workload management for the exits. The exit manager handles both IBM-supplied and custom user exits.

IMS Connect user message exits perform operations such as translation between IRM (header of TCP/IP frames) and OTMA protocols and between ASCII and EBCDIC character code sets. The use of assembler message exits ensures that IMS Connect has both high performance and flexibility.

You do not need to modify IMS Connect exits to work with IMS Connect Extensions. Figure 9-13 on page 191 displays the functions that the exit manager performs.

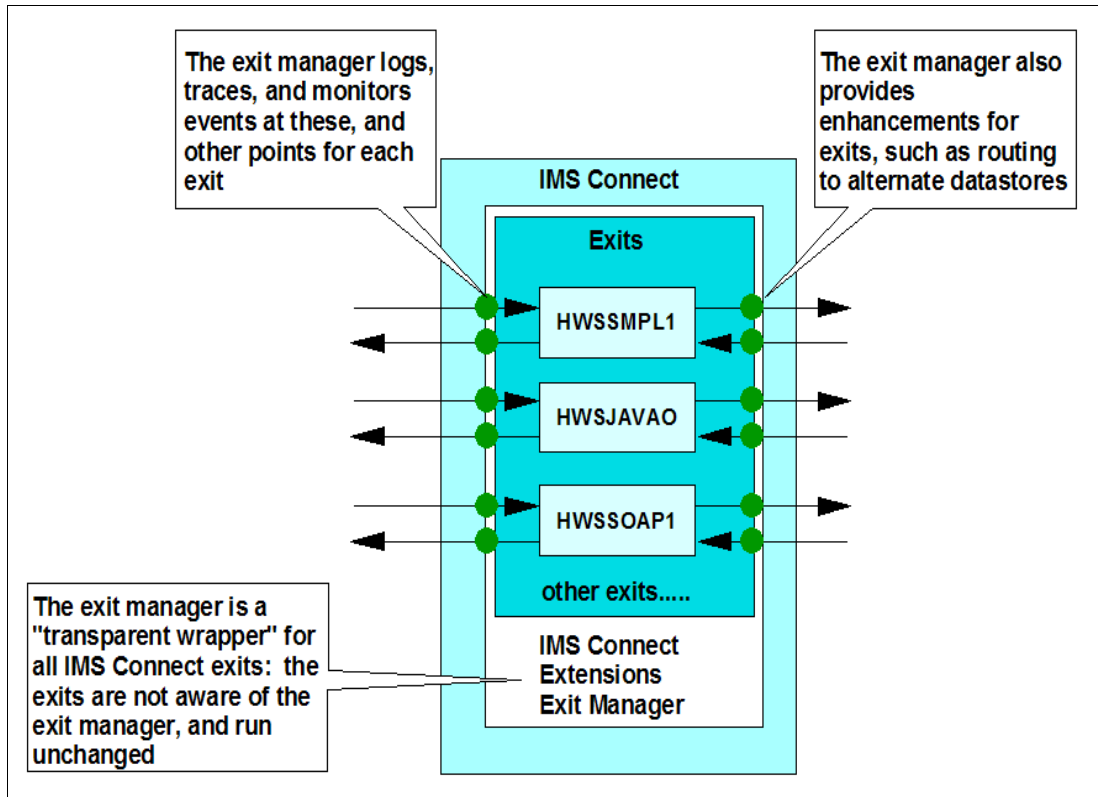


Figure 9-13 IMS Connect Extensions wrapping the user exits

► Event collector

IMS Connect Extensions collects information directly from IMS Connect through the exit manager. The exit manager initializes the event collector to collect and record IMS Connect events. This information about activity in IMS Connect is written to journals, although the activity that occurs within OTMA and IMS is recorded in the IMS logs. Other IBM IMS tools such as the IMS Problem Investigator can combine the data to provide an end-to-end picture of activity for a given transaction.

Figure 9-14 on page 192 presents an overview of the event collection. There are three phases in the event collection process:

- Pre-OTMA: A message is read by IMS Connect from the Socket, transformed, handled by the user exit, and passed to OTMA.
- OTMA processing in IMS
- Post-OTMA: The response returns from OTMA to IMS Connect and is transformed by the user exit and written on the socket.

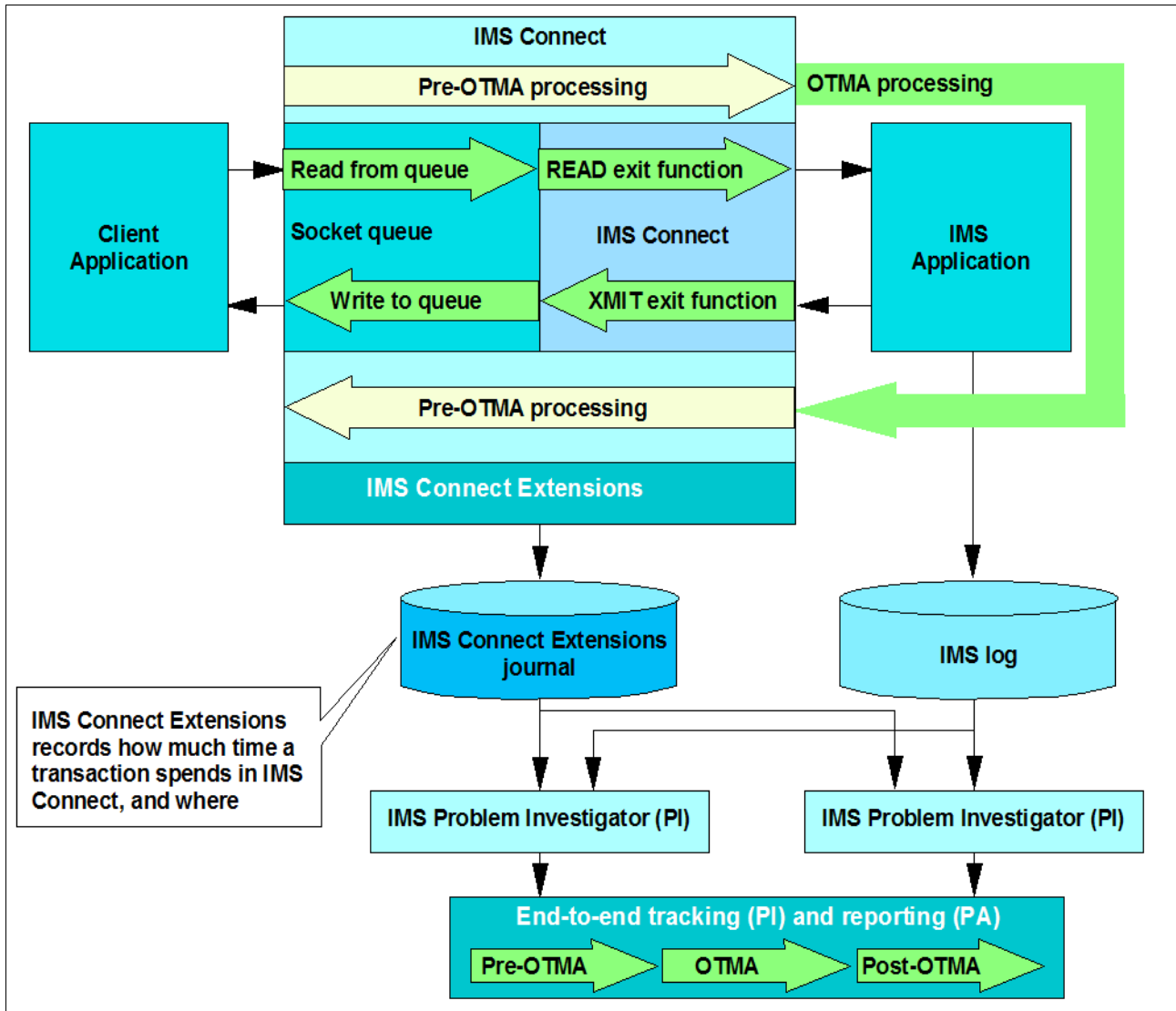


Figure 9-14 IMS Connect Extensions even collection

There are two types of event records that are collected:

- Status event:

A Connect status event identifies a change in the status of your IMS Connect system. For example, a resource becoming available or unavailable, a socket becoming accepted for input by a port task, or a journal switch.

- Message related event

For non-persistent sockets, each incoming message is assigned a unique Event Key and every event associated with the processing of the message has the same Event Key.

For persistent sockets, all messages are assigned the same Event Key. All events that are associated with the processing of all messages for the duration of the socket have the same Event Key.

IMS Connect Extensions associates message related event records with each other so that the sequence of events and event times can be identified and reported. The event

records are associated with each other using the Event Key. This allows IMS Connect Extensions to group together event records in the sequence they occur.

Continuing with the main components of IMS Connect Extensions:

- ▶ **Status monitor agent**
Provides real-time statistics on message processing activity for a system, its exits, data stores, and ports. Status monitor statistics are available through the console or publisher interface (see “Publisher interface” below).
- ▶ **Active session agent**
Provides real-time tracking of active TCP/IP and LOCAL port sessions. Active session information is available through the console or publisher interface. Through the console, you can view details of running sessions and stop hung sessions.
- ▶ **Publisher interface**
Provides monitoring real-time data about the IMS Connect system and tasks running in an IMS Connect region. The publisher interface supports the IMS Connect monitoring function of OMEGAMON for IMS on z/OS.
- ▶ **Active and archive journals**
These files contain the IMS Connect event data recorded by IMS Connect Extensions. Active journals are those journals the event collector directly writes to. Archive journals are sequential data sets or GDGs, on disk or tape, containing data archived from active journals. IMS Connect Extensions journals can be analyzed by other IBM IMS tools, such as IMS Problem Investigator and IMS Performance Analyzer, to provide debugging, performance analysis, and auditing information. IMS Connect Extensions includes a print utility that can format journal data and write it to a data set.
- ▶ **Batch utilities**
IMS Connect Extensions includes batch utilities that provide additional features, such as log formatting and batch administration.

9.9.2 Integrating IMS Connect Extensions with IMS Connect

IMS Connect Extensions is an extension of IMS Connect, and its additional agent code runs in the IMS Connect address space in APF authorized mode.

Configurations can be maintained through the ISPF dialog or the Batch Definition utility. After the ISPF dialog is installed, one possibility is to start with an empty repository and then add the configuration descriptions for both IMS Connect and Extensions.

Example 9-6 displays the primary ISPF menu for IMS Connect Extensions.

Example 9-6 IMS Connect Extensions 2.1 - Primary Menu

```
                    IMS Connect Extensions 2.1 - Primary Menu
Option ==>>>

0 Profile           Customize your dialog profile
1 Definitions       Display or maintain IMS Connect Extensions definitions
2 Operations        System Monitor and Control Facility
X Exit             Exit IMS Connect Extensions

Definitions repository . . 'CEX.V2R1M0.CEXREPOS'
```

Taking the *Definitions* path, you see the different elements that can be maintained, as presented in Example 9-7.

Example 9-7 Definitions menu

Definitions

Option ==>

1	System Definitions	Specify definitions for IMS Connect systems
2	User Exits	Specify User Message Exits
3	Datastores	Define datastore processing options
4	Datastore Groups	Group datastores for collective control
5	Affinity Lists	Associate datastores for Affinity processing
6	Applications	Group transactions for Application control
7	Transactions	Define transaction processing options
8	System Groups	Group IMS Connect systems for collective control

Taking the *System Definitions* path, you see in Example 9-8 that one system, IMS Connect "IMSLCONN", was defined.

Example 9-8 System Definitions menu

System Definitions

Command ==> Row 1 from 1
Scroll ==> PAGE

Filter . . . (Blank or pattern)

Enter "/" to select action

/	Name	Description	----- Changed -----	ID
	IMSLCONN	HWS Take up	2008/07/09 23:13:38	RC67

From this panel, we can add new system definitions or select existing ones. Selecting "/" against the IMSLCONN name results in being able to choose options EDIT or VIEW or DELETE. Example 9-9 presents the output when the "VIEW" option is selected.

The settings that you see are related to the active and installed extensions of IMS Connect. The "console settings" indicate how the ISPF dialog can connect to the IMS Connect Extensions console. This information is read by an IMS Connect when it starts and allows the activation of the console listener port.

Example 9-9 Output of the View option for entry IMSLCONN

VIEW System Definition End of data

Command ==>

Name : IMSLCONN

Description . . HWS Take up

Console settings:

Port number 9998	Message recall count 800
Host name LOCALHOST	

Product features:

- / Activate IMS Connect Extensions
- / Activate Event Collection

```

Collection level . . . . 4          Log Record number . . AO (A0-FF)
  Activate Publisher API          Maximum clients . . . 0 (0-99)
Enter "/" to edit Journal Data Set template details
  Active Data Set 'CEX.ACTIVE.&ID'
  Archive Data Set 'CEX.ARCHIVE.&ID..&JFIRST'

/ Activate Commands
  Activate Access Control          Security applid . . .
  Activate PassTicket Generation

/ Activate Advanced Features
  Activate Pacing
  Interval count . . . . 3
  Warning threshold . . 0          Reject threshold . . 0

  Activate Security
  Activate ACEE Cache             Ageing interval . . . 60 (Min)
  Activate IMS Connect validation Security class . . .

  Activate Transaction Routing
  Define Applications for IMSLCONN
  Activate Workload Balancing

/ Activate Statistics Collection

Related definitions:
  Edit IMS Connect configuration

```

Taking the *Operations* path from the “Primary Option menu”, you observe the panel that is displayed in Example 9-10. One IMS Connect system is defined and active. You also see that events collection is ON.

Example 9-10 IMS Connect definition

```

                                Operations - Systems View          Row 1 from 1
Command ===>                                Scroll ===> PAGE

View . . . 2  1. Groups  2. Systems

Filter . . .          Exclude inactive systems

/ System  Status  VRM  Description          Events  Journal  Member
  IMSLCONN ACTIVE  210  HWS Take up          ON     P03

```

When you select a defined IMS Connect, you get a list of the available Line Actions, as displayed in Example 9-11.

Example 9-11 Line Actions

```

                                Operations - Systems View          Row 1 from 1
Command ===>                                Scroll ===> PAGE

View . . . 2  1. Groups  2. Systems

```

```

Filter . .                Exclude inactive systems

/ System Status VRM Description                      Events Journal Member
/ IMSLCONN ACTIVE 210 HWS Take up                     ON      P03
Esxxxxxxxxxxxxxxxxxxxxx Line Actions sxxxxxxxxxxxxxxxxxxxxxN *****
e
e
e Select by number or action code then press Enter.    e
e 1. Display status monitor... (SM) e
e 2. Display active sessions... (AS) e
e 3. Issue IMS Connect Extensions commands... (CX) e
e 4. View message log... (L) e
e 5. Issue IMS Connect commands... (SH) e
e 6. Manage Publisher clients... (PU) e
e 7. Set tracing by resource... (TR) e
e 8. Start recorder trace (RS) e
e 9. Stop recorder trace (RP) e
e 10. Switch Journal (J) e
e 11. Stop system (P) e
e 12. Stop system (with Force) (F) e
e 13. Exclude system (X) e
e
e F1=Help      F3=Exit      F6=Resize      F7=Backward e
e F8=Forward   F12=Cancel
DsxxxxxxxxxxxxxxxxxxxxxM

```

Although most of the commands are self-explanatory, you still have the “Help” function on the panel available. For more information, refer to:

IBM IMS Connect Extensions for z/OS v2.1 User’s Guide, SC19-1163

9.9.3 Using IMS Connect Extensions with other IMS tooling

In this section, we discuss how to use IMS Connect Extensions with other IMS tooling.

Reporting and analysis with the IMS Performance Analyzer

The IMS Performance Analyzer for z/OS is a performance analysis and tuning aid for DB and TM systems for IMS Version 8 and above. It processes log, monitor, and IMS Connect event data to provide comprehensive reports for use by managers, database administrators, communications administrators, and system programmers to analyze and tune their IMS systems.

IMS Performance Analyzer provides a comprehensive set of reports from the IMS Connect performance and accounting data collected by IMS Connect Extensions. The reports provide a summary and detailed analysis of IMS Connect transaction transit time, resource usage, and resource availability:

► Transaction Transit Reports

These reports provide performance statistics to measure the performance of your IMS Connect transactions. Transaction transit (response) time is broken down into its components; input, processing (by OTMA), acknowledgement from the client, and output. They can help identify any bottlenecks in transaction flow, and are used for monitoring system performance, gathering diagnostic information, and tuning IMS.

- ▶ Resource Usage Reports

These reports contain detailed and summary information on the use and availability of various IMS Connect resources, including TCP/IP ports and TPIPEs.

- ▶ Trace Reports

The trace provides a list of transactions, each with detailed information about every event in the life of that transaction.

Reporting and analysis with the IMS Problem Investigator

IMS Problem Investigator is an intelligent log analysis tool that can assist for tasks, such as debugging, performance tuning, tracing, and creating audit trails.

IMS Problem Investigator merges information in the IMS logs with information collected by IMS Connect Extensions, which allows you to dissect individual transactions as they progress from TCP/IP, to OTMA, to IMS, and back. Like the IMS Performance Analyzer, IMS Problem Investigator automatically selects IMS Connect Extensions archive journals and IMS logs.

With the IMS Problem Investigator you can:

- ▶ View formatted logs with detailed field descriptions.
- ▶ Navigate to an exact point in time within a log file.
- ▶ Investigate specific problem areas. For example for transaction, database, security, or checkpoint processing.

OMEGAMON for IMS on z/OS

Using IMS Connect Extensions you gain a real-time graphical view of TCP/IP sessions and performance throughput using OMEGAMON.

Refer to the *OMEGAMON XE* for Mainframe Networks Library available at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon.mn.doc/sc32-1924-0009.htm>

9.10 IMS Connect as a database router

Besides the traditional access to IMS databases through the CCTL interfaces, such as CICS, another IMS component, Open Data Base Access (ODBA), is available.

ODBA can be used by DB2 stored procedures and WebSphere Application Server through its DataBase Resource Adapter. The limitation however is that the ODBA caller and IMS must be in the same LPAR. As presented in Figure 9-15 on page 198, a solution was devised to allow a remote WebSphere Application Server to access IMS/DBCTL through a local WebSphere Application Server on the same z/OS LPAR as IMS.

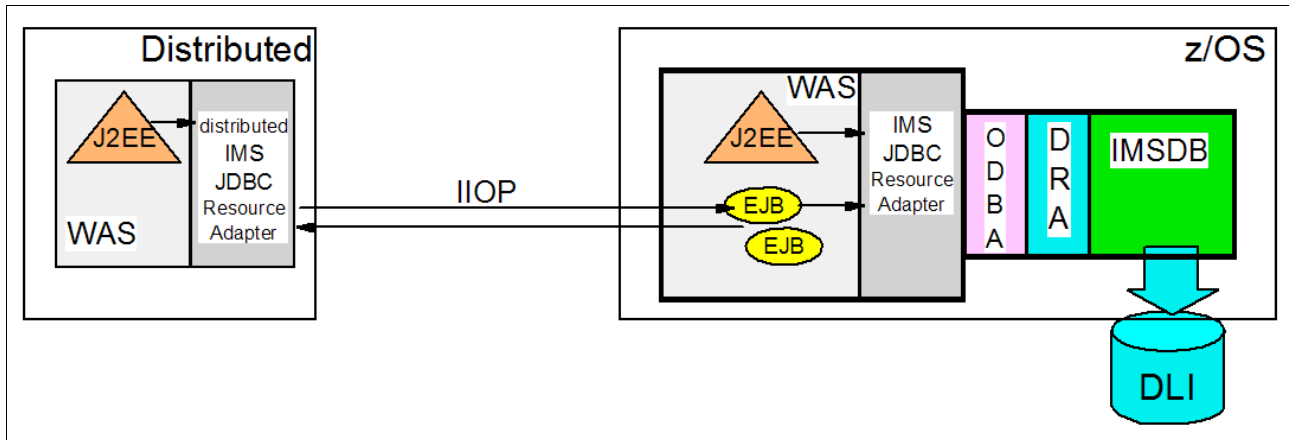


Figure 9-15 Distributed access to IMS data through ODBA

The communication occurs over a *remote DB resource adapter*, acting as an EJB client for a local Session EJB, which over the *local DB resource adapter*, talks with ODBA. But with IMS Version 11, IBM offers additional facilities, which support:

- ▶ A local WebSphere Application Server is not required anymore to remotely access local IMS data stores.
- ▶ Through IMS Connect, we can access IMS data bases from other z/OS LPAR's than the local one.

Supplied in IMS Version 11, Open Database Manager (ODBM) is a separate address space that operates under the IMS Common Service Layer (CSL), and manages the routing of database access calls. IMS Connect then routes the response from ODBM back to the client.

IMS Universal drivers broker all communications between your Java applications and the IMS subsystem by sending DRDA messages using TCP/IP protocols. IMS Connect serves as the TCP/IP server and router through the use of an internal routing table to find the ODBM that supports the IMS alias that is requested by the client and passes the request to ODBM.

As a part of the support for IMS Universal drivers, IMS Connect supports two-phase commit for global RRS transactions. The following IMS Connect exit routines support the IMS Universal drivers:

- IMS Connect ODBM Security exit routine (HWSAUTH0)
- IMS Connect Routing exit routine for ODBM (HWSROUT0)

Example 9-12 shows an example of a HWSCFGxx member describing an ODBM connection.

Example 9-12 HWS Connect example for ODBM

```

*****
* IMS Connect example for IMS Universal drivers and DRDA client support
*****
HWS (ID=HWS1,RACF=N,XIBAREA=20)
TCP/IP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999),MAXSOC=2000,TIMEOUT=8888,EXIT=(HWSSMPL0))
DATASTORE (ID=IMSA,GROUP=GROUPA,MEMBER=HWS1MEM,TMEMBER=IMSAMEM,DRU=HWSYDRU0)
ODACCESS (DRDAPORT=(ID=1111,KEEPAV=5,PORTTROT=50),DRDAPORT=(ID=2222,KEEPAV=10,PORTTROT=500),
ODBAUTOCONN=Y
IMSPLEX=(MEMBER=IMSPLEX1,TMEMBER=PLEX1),
*****

```


Details about IMS Connect for DB are in Chapter 15, “IMS Open Database and Universal Drivers” on page 321.

9.11 IMS Connect use in a Sysplex and IMSplex

First, we define a few important terms.

Sysplex A sysplex environment is a set of z/OS systems that communicate and cooperate with one another through certain multisystem hardware components, and software services to process workloads.

IMSplex An IMSplex is one or more IMS control regions, managers, or servers that work together as a unit.

The OTMA function uses XCF APIs to define a transaction pipe (TPIPE) connection between an OTMA client (IMS Connect) and the IMS TM OTMA server function. The TPIPE supports full-duplex protocol. IMS Connect must be a member of a XCF group that also includes the IMS subsystems it wants to communicate with.

Now, we discuss the use of IMS Connect and the OTMA super member feature.

9.11.1 IMS Connect support for z/OS Sysplex Distributor

In a z/OS Sysplex Distributor environment, incoming messages are typically distributed among multiple instances of IMS Connect to balance the workload and increase availability. In such an environment, client applications have no control over which instance of IMS Connect receives their input messages and which IMS Connect receives subsequent requests for asynchronous output.

IMS Connect with OTMA provides several features to support operating in such an environment, such as rerouting asynchronous output to an alternate Tpipe, sharing asynchronous output by using an OTMA super member tpipe, retrieving output from an alternate tpipe queue associated with another client, and purging undeliverable output.

OTMA super member

The OTMA super member function is specifically designed to support multiple instances of IMS Connect in a z/OS Sysplex Distributor environment. The OTMA super member function allows for sharing of “asynchronous commit-then-send output (CM0)” by Hold-queue-capable OTMA clients, such as IMS Connect.

The OTMA super member function manages all of the asynchronous CM0 output of all of its participating OTMA clients by using a common output queue. Any participating hold-queue-capable client can then retrieve the CM0 messages on the super member output queue by issuing its own RESUME TPIPE call, regardless of which client the CM0 output was originally destined for. The messages on the super member queue are retrieved on a first-in-first-out basis, without regard to which instance of the OTMA client originated the output or which instance of the OTMA client issued the RESUME TPIPE call.

You can purge the output, reroute the output, or share the output among multiple OTMA clients by using an OTMA super member.

To use the OTMA super member in a configuration that includes multiple IMS systems, you must enable shared queues for all those IMS systems. If a shared queues back-end IMS creates ALT-PCB output in the super member-enabled environment, the output can be

retrieved from any front-end IMS with an OTMA client in the super member set. Figure 9-16 presents a pictorial overview of super member support.

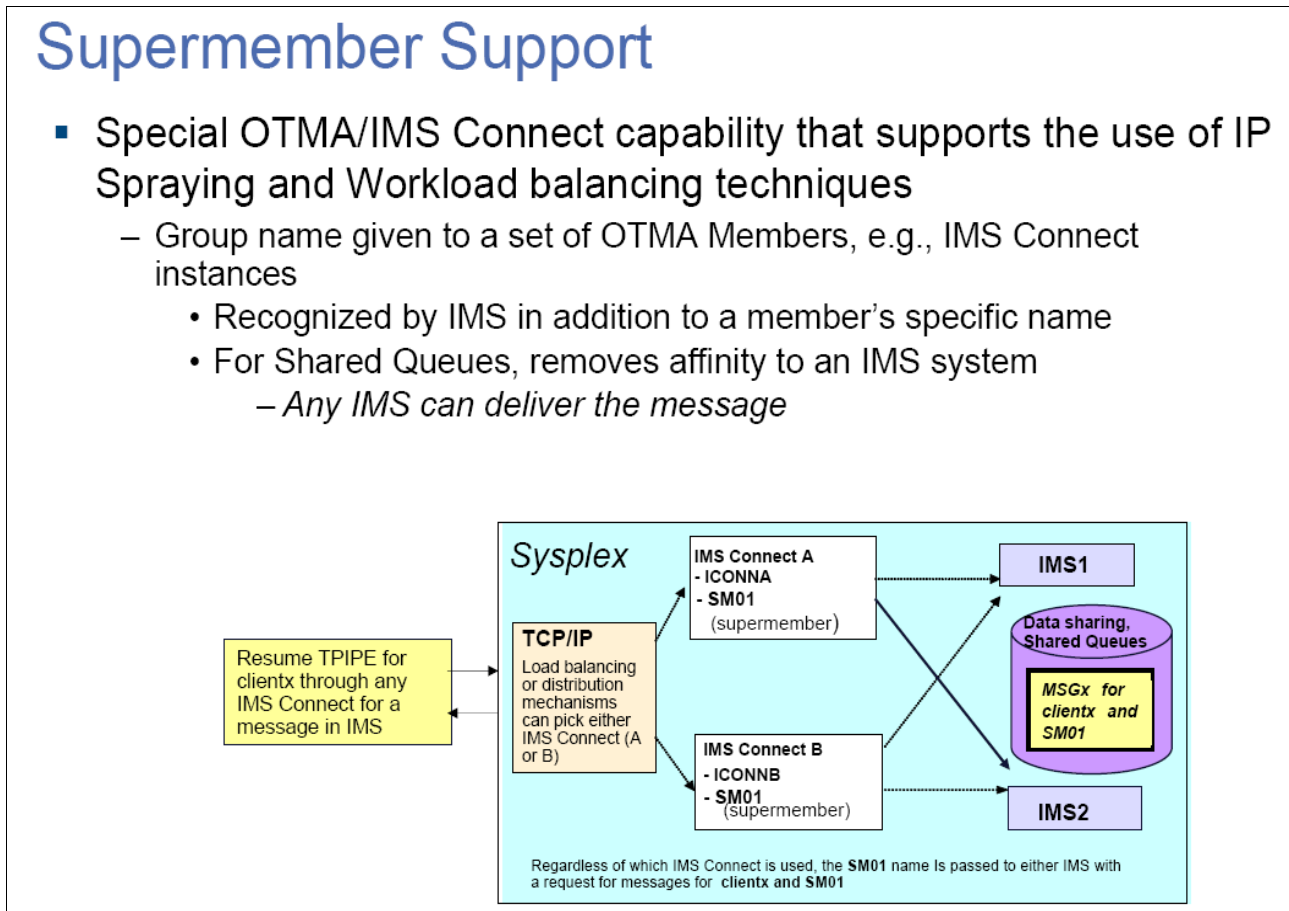


Figure 9-16 Super member support

If there is only one IMS system with multiple OTMA clients, shared queues support is not required. To activate the super member function, specify a one to four character super member name in the SMEMBER parameter of the HWS configuration statement in the IMS Connect configuration member (HWSCFGxx).

9.11.2 IMS Connect support for IMSplex and the IMS Control Center

IMS Connect can send and receive IMS Operations Manager (OM) commands and response string messages between an IMS Control Center client and the OM in an IMSplex by using the IMS Structured Call Interface (SCI). Figure 9-17 on page 201 displays the components that are associated with the IMS Control Center and IMS Connect.

SCI allows IMSplex members to communicate with one another. The communication between IMSplex members can happen within a single z/OS image or among multiple z/OS images. Individual IMS components do not need to know where the other components reside or what communication interface to use.

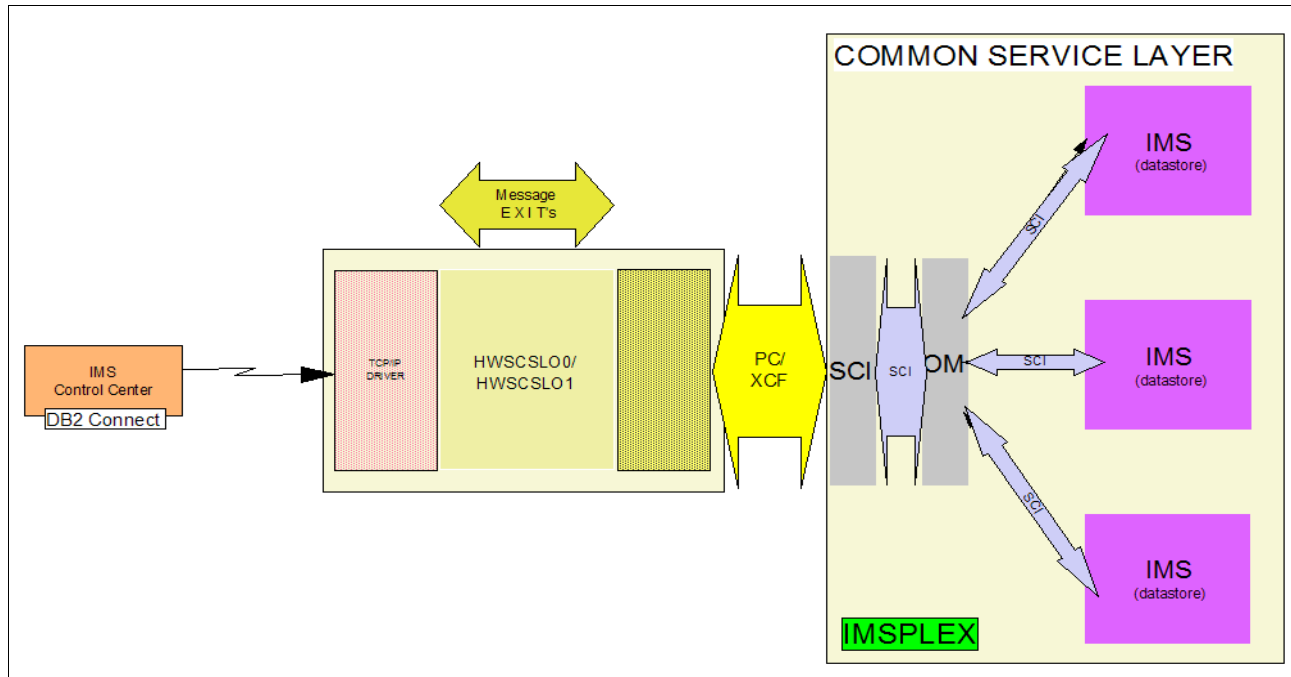


Figure 9-17 IMS Control Center environment

IMSpIex support sends IMS command string messages directly from a client (for example, the IMS Control Center supplied TCP/IP client) to a selected OM within an IMSpIex. One or more IMSpIexes can be defined to IMS Connect to receive DB2 Control Center client command messages.

The IMS Control Center exit routines (HWSCSLO0 and HWSCSLO1) are designed specifically for IMSpIex support.

9.11.3 IMS Connect support for IMSpIex and shared queues

Operating in a shared-queues environment allows multiple IMSs in a sysplex environment to share IMS message and EMH message queues. The shared-queues environment distributes processing loads between the IMSs in the IMSpIex. Transactions that are entered on one IMS can be made available on the shared queues to any other IMS that can process them. The shared-queues environment distributes processing loads between the IMSs in the IMSpIex.

9.12 Conclusion

IMS Connect is the gateway through which all new external connections are passing, both inbound and outbound. This solution is clean because we utilize the OTMA links with IMS over a XCF and SCI mechanism. The domains that IMS Connect supports are:

- ▶ Inbound (callin) transaction support
- ▶ Outbound (callout) to Web Services and EJB's in both synchronous and asynchronous modes
- ▶ Entry point for Single Point of Control(SPOC) from the IMS Control Center in DB2Connect
- ▶ Inbound DB router for DLI access



The IMS SOAP Gateway

IMS SOAP Gateway is a light weight Web Services solution that enables IMS applications to inter-operate in a service-oriented architecture environment without the need of a full featured application server (for example, Java EE server). Through the use of Simple Object Access Protocol (SOAP), IMS SOAP Gateway provides services that are independent of platform, environment, application language, or programming models.

In this chapter, we provide a high-level understanding of IMS SOAP Gateway and how IMS SOAP Gateway fits into the overall IMS architecture. Also, we introduce several new enhancements that are now available with IMS SOAP Gateway.

10.1 IMS SOAP Gateway implementation overview

IMS SOAP Gateway enables your IMS applications to perform as Web Service providers and consumers.

Different types of client applications, such as Microsoft .NET, Java, and third-party applications can submit SOAP requests into IMS. In conjunction with the IMS Connect XML adapter, you can enable your IMS application to become a Web Service without changing the back-end IMS application.

IMS SOAP Gateway also enables your IMS application as a Web Service consumer. Your IMS applications can make a callout request to access external Web Service providers and get responses back either synchronously or asynchronously.

IMS SOAP Gateway is compliant with the industry standards for Web Services, including SOAP/HTTP 1.1 and Web Services Description Language (WSDL) 1.1. This compliance enables your IMS assets to interoperate openly with various types of applications.

The IMS SOAP Gateway tool has several different components. The components are:

- ▶ **Connection bundle:** This specifies the connection and security properties between IMS SOAP Gateway, IMS Connect, and IMS. In conjunction with the correlator file, this information is passed to IMS Connect from IMS SOAP Gateway. The connection bundle is generated by the IMS SOAP Gateway Deployment utility. Property options for the connection bundle are:

- Connection bundle name
- IMS Connect host name
- IMS Connect port
- IMS Connect Datastore
- User ID
- Password
- Groupname

For SSL support, the settings also include:

- SSL Keystore name
- SSL Keystore password
- SSL Truststore name
- SSL Truststore
- SSL Encryption level

For callout support, it is necessary to indicate the TPIPE name(s).

Example 10-1 shows a sample connection bundle.

Example 10-1 Connection bundle example

```
<root>
<conn>
<connBundleName>connbundle1</connBundleName>
<connHostName>your.host.name</connHostName>
<connPortNum>yourPortNumber</connPortNum>
<connDataStoreName>yourDataStoreName</connDataStoreName>
<connUserID></connUserID>
<connPassword></connPassword>
<connGroupName></connGroupName>
<sslKeystoreName></sslKeystoreName>
<sslKeystorePasswd></sslKeystorePasswd>
```

```

<sslTruststoreName></sslTruststoreName>
<sslTruststorePasswd></sslTruststorePasswd>
<sslEncrypType></sslEncrypType>
</conn>
</root>

```

- ▶ **Correlator file:** An xml file that can be generated by RDz or the IMS SOAP Gateway deployment utility. It specifies transaction, runtime properties, and information that the IMS SOAP Gateway needs to match incoming requests to the appropriate back-end IMS application. After IMS Connect receives the user specifications from the correlator file it uses the specification to know how long to wait, and so on. It also identifies the Connection bundle. Property options for the correlator are:

- Connection bundle name
- Socket time-out
- Execution time-out
- LTERM name
- IMS transaction code
- Program Name
- XML Adapter type
- XML converter name

Additionally, the properties that support IMS applications as Web Service consumers are:

- Callout connection bundle name: The name of the callout connection bundle that contains the connection and security properties that are used to connect to IMS.
- Callout Web Service WSDL filename: The name of the WSDL file of the Web Service to which an IMS application makes a callout request.
- Callout Web Service invocation time-out: The amount of time in milliseconds that IMS SOAP Gateway must wait to receive a response from the external Web Service.
- Callout service name: The service name specified in the WSDL file.
- Callout operation name: The operation name specified in the WSDL file.

Example 10-2 displays an example correlator file.

Example 10-2 Correlator file example for IMS SOAP Gateway

```

<?xml version="1.0" encoding="UTF-8"?>
<correlator:correlator xmlns:correlator="http://www.ibm.com/IMS/Correlator">
  <correlatorEntry operationName="IMSPHBKOperation"
    serviceName="IMSPHBKService">
    <SOAPAction>IMSPHBK</SOAPAction>
    <adapterType>IBM XML Adapter</adapterType>
    <converterName>IMSPHBKD</converterName>
    <socketTimeout>0</socketTimeout>
    <executionTimeout>0</executionTimeout>
    <ltermName></ltermName>
    <adapterType>IBM COBOL XML Adapter</adapterType>
    <connectionBundleName>conn</connectionBundleName>
    <inboundCCSID>1208</inboundCCSID>
    <hostCCSID>1140</hostCCSID>
    <outboundCCSID>1208</outboundCCSID>
    <inboundTPIPName></inboundTPIPName>
    <trancode></trancode>
    <calloutConnBundleName>
      </calloutConnBundleName>
    <calloutWSDL></calloutWSDL>
  </correlatorEntry>
</correlator>

```

```

    <calloutWSTimeout></calloutWSTimeout>
  </correlatorEntry> </correlator:correlator>

```

- ▶ WSDL file: An XML document that describes a Web Service. WSDL files are used by others (for example, the client that invokes the service) to discover the service, and to understand how to invoke the service. It specifies the location of the service and the operations that the service exposes. WSDL files are generated by RDz.
- ▶ Web Service: These are services; usually some combination of programming and data, that are made available for web users or other web-connected programs.
- ▶ Server properties file: This is used to configure the IMS SOAP Gateway Server runtime environment. It is generated by the SOAP Gateway deployment utility.
- ▶ Deployment utility: This utility is provided by IMS SOAP Gateway. It generates the connection bundle, the correlator file, and the server properties file. The Deployment utility runs in the Windows DOS Command Prompt.

Figure 10-1 presents a panel shot that is associated with the main panel of the IMS SOAP Gateway deployment utility.

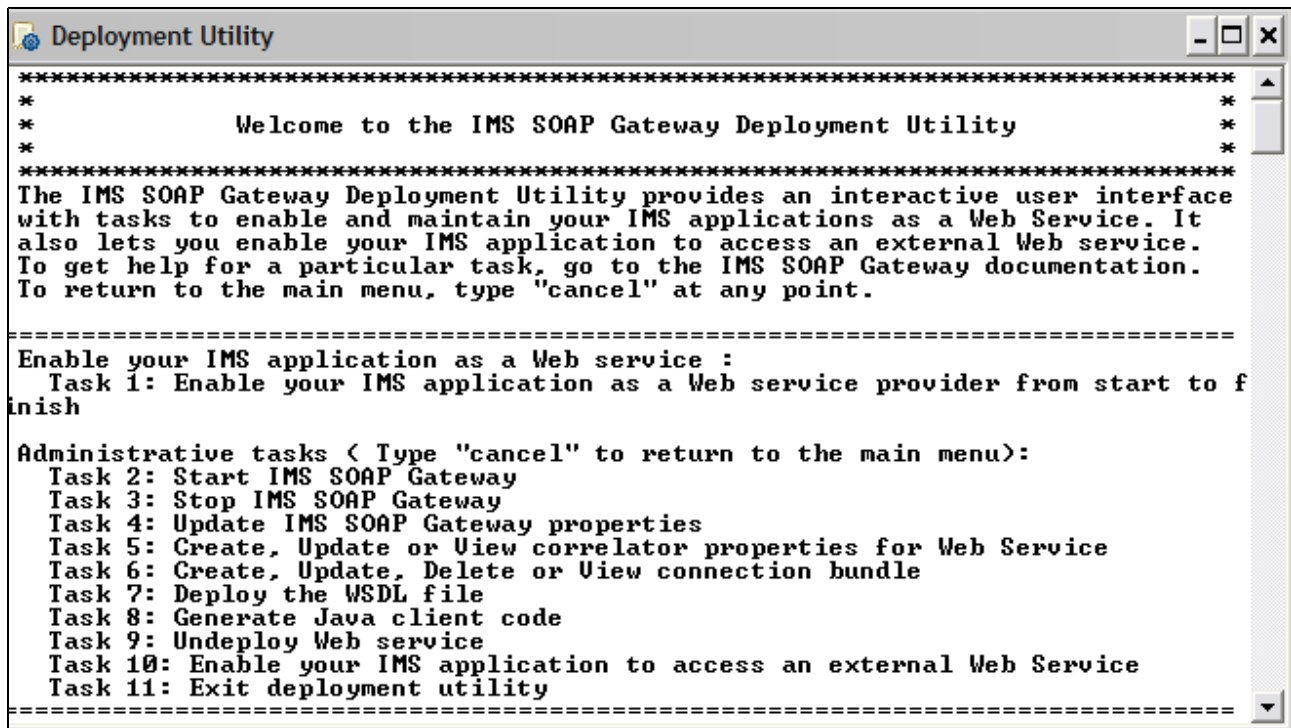


Figure 10-1 Screen shot of the IMS SOAP Gateway Deployment Utility

- ▶ IMS Connect XML Adapter: The IMS Connect XML Adapter receives XML requests from IMS Connect. The XML Adapter then sends the XML message to a XML conversion function that can transform the XML request message into an application data format that the IMS COBOL application expects.
- ▶ IMS Connect XML Converter: The XML converters called by the XML Adapter are specific to each COBOL application. The converters can be generated by Rational Developer for System z using the COBOL copybook of the COBOL application. The IMS SOAP Gateway specifies to IMS Connect which XML converter to use for data transformation for the intended IMS COBOL application. IMS SOAP Gateway also supports PL/I.

Note: Refer to Table 10-1 on page 223 to view a table of IMS SOAP Gateway features and a matrix on compatibility of IMS SOAP Gateway, IMS, IMS Connect, and Rational Tooling levels.

We now provide examples of the flow of control that occurs with IMS SOAP Gateway solutions.

10.2 IMS SOAP Gateway and your IMS applications

IMS SOAP Gateway can enable your IMS applications as both a consumer and provider. Figure 10-2 shows the flow of control that is associated with an IMS application as a service provider.

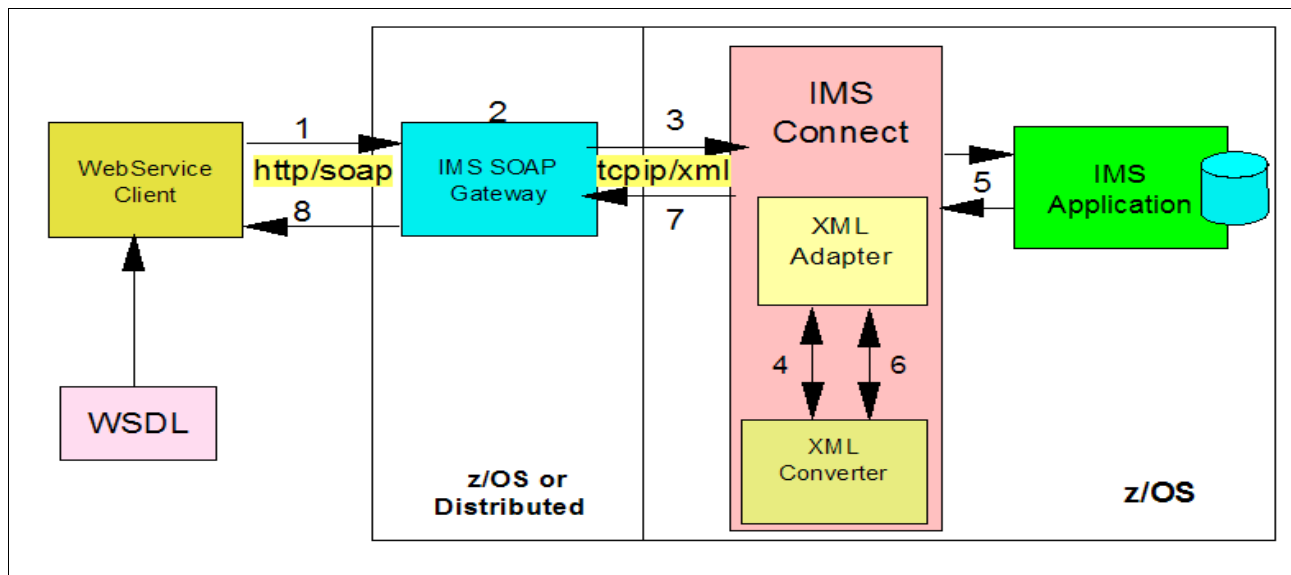


Figure 10-2 Flow of control with an IMS application as a Web Service provider

The flow of control that is associated with an IMS application as a service provider:

1. The Web Service client application sends a SOAP message to IMS SOAP Gateway.
2. IMS SOAP Gateway processes the SOAP header (XML) and retrieves the appropriate correlation and connection information for the input request.
3. IMS SOAP Gateway sends the input XML data to IMS Connect using TCP/IP after adding the appropriate IMS Connect header.
4. IMS Connect calls the XML Adapter which in turn calls the XML Converter to perform the XML to IMS application format transformation.
5. IMS Connect then sends the message to IMS for further processing.
6. IMS Connect calls the XML Adapter to perform the transformation of the IMS application format response into XML.

7. IMS Connect sends the output XML message back to IMS SOAP Gateway using TCP/IP.
8. IMS SOAP Gateway wraps a SOAP header on the output message and sends it back to the client application.

IMS SOAP Gateway can also be used in environments where IMS applications are a Web Service consumer, requesting services. 10.4, “Asynchronous Callout with SOAP Gateway” on page 209 and 10.5, “Synchronous Callout with IMS SOAP Gateway” on page 216 detail these processing flows.

10.2.1 XML-to-bytes and bytes-to-XML

The XML Adapter supports translation between XML and IMS messages. IMS Connect’s XML Adapter was created to give support to IMS Connect clients such as IMS SOAP Gateway. For inbound messages, IMS Connect invokes the XML Adapter to translate the message for IMS, remove XML tags, and if necessary convert from UNICODE to EBCDIC. For outbound flows, IMS Connect invokes the XML Adapter to prepare a XML message and creates the XML tags. Also if necessary, it converts the message from EBCDIC to the appropriate UNICODE encoding schema.

Rational Developer for System z (RDz) supports the generation of XML converters for COBOL or PL/I applications using COBOL copybooks or PL/I include files. The XML Adapter support in IMS Connect in conjunction with the generated RDz XML converters, facilitates the conversion of XML transactional requests into byte stream application data structures, and vice-versa. Each IMS application expects its messages to be in a certain data structure, therefore one XML Converter is needed for each IMS application.

If you choose to handle the data transformation in your application without utilizing the IMS Connect XML Adapter, you obviously do not need to invoke the XML Adapter. In this case, the incoming XML message is sent to IMS Connect and then to the IMS application. The same is true in reverse. The IMS application creates a XML output message which is sent to IMS Connect, then directly to IMS SOAP Gateway, and last to the Web Service client. The problem with not using the XML Adapter is that the IMS application needs to handle converting byte data into the XML data structure because the IMS transaction code cannot be processed in XML format.

Note: IMS Soap Gateway must be at a minimum level of Version 9.2 to take advantage of the XML Adapter in IMS Connect.

10.3 What is new in IMS SOAP Gateway

Because IMS Version 9 was originally generally available in the fall of 2005, IMS SOAP Gateway released two versions (9.2.1 and 10.1), and its capabilities grew tremendously. The key new features are:

- ▶ Platform support for z/OS and zLinux
- ▶ Security enhancements, such as SSL/HTTPS and Web Services security
- ▶ IMS Connect XML Adapter support
- ▶ Accessibility enablements
- ▶ Asynchronous and synchronous Callout support

10.4 Asynchronous Callout with SOAP Gateway

In the recent years, there is an increasing need for core IMS business functions to access new business logic and data running outside the IMS environment. There is interest in transforming IMS applications into a Business-to-Business (B2B) clients, not just components within a B2B server.

IMS Version 10 adds application asynchronous callout support to a Web Service through IMS SOAP Gateway. IMS Callout refers to the ability of an IMS application to act as a client to invoke an application that is external to the IMS environment (for example, an application running in the distributed platform).

To support IMS Asynchronous Callout to Web Services, IMS SOAP Gateway implements the IMS Resume TPIPE protocol internally and manages a resume TPIPE loop inside the IMS SOAP Gateway server to continually retrieve asynchronous messages that were inserted from an alternate TPPCB by the IMS application. After the callout request message is received, IMS SOAP Gateway processes the request, finds the Web Service identifier in the callout request message and matches it with the deployed Web Services to locate and invoke the desired Web Service. The invoked Web Service either performs the task and ends or sends the callout response message back to IMS to start a new IMS transaction, as shown in Figure 10-3.

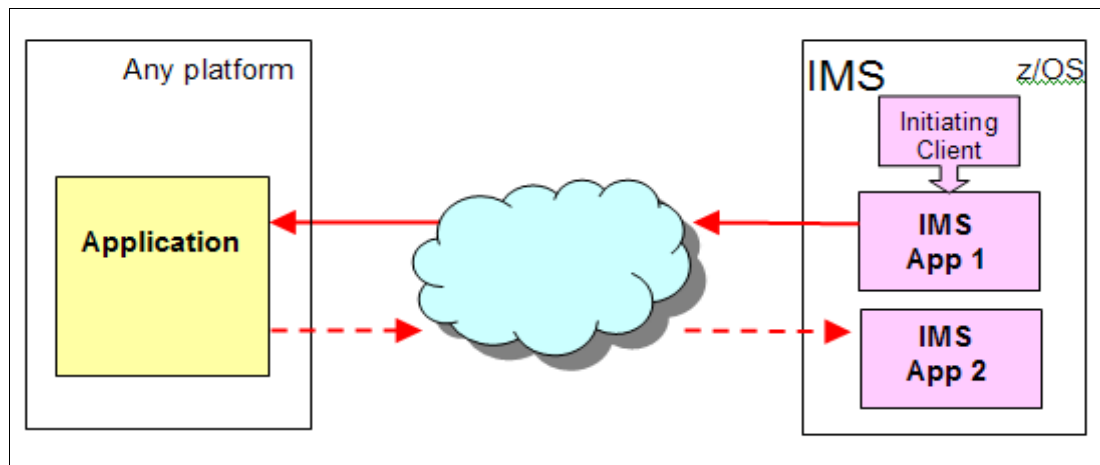


Figure 10-3 Asynchronous Callout flow

The advantage of using asynchronous callout is that it does not hold up the region, but it does require the IMS application to be designed in a way that is able to handle the output from the external application (if any) in a different IMS application instance.

10.4.1 Invoking the Web Service operation

There are two types of Web Service operations for asynchronous callout: the one-way invocation and the request-response invocation. The one-way invocation corresponds to the one-way port type and the request-response invocation corresponds to the request-response port type of the Web Service operation. In the one-way invocation, no response is expected from the Web Service. In the request-response invocation, a response message is synchronously returned back to the caller.

If the operation is a one-way, it sends out the request message by taking the XML message as the payload data but does not wait for the response to come back. If the operation is a request-response, IMS SOAP Gateway waits for the response data to synchronously come back from the same SOAP/HTTP connection.

After the request message is sent successfully, IMS SOAP Gateway sends an ACK (positive acknowledgement) internally to IMS Connect to indicate the callout request message is received and processed such that it can be removed from the TPIPE. In the case of the request-response request, the ACK is sent back to IMS Connect after the request is sent successfully to the Web Service but before IMS SOAP Gateway starts waiting for the response message from the Web Service.

If the send fails or a processing error occurred before invoking the Web Service, a NAK (negative acknowledgement) with reroute command is sent to IMS Connect such that the callout request message is removed from the callout TPIPE and be rerouted to an error queue.

10.4.2 Returning the callout response message to IMS

In a request-response invocation, a callout response message is returned by the Web Service to IMS SOAP Gateway. For asynchronous callout, this response message is sent back to the IMS system by invoking a new IMS transaction, which we call Appl2.

When IMS SOAP Gateway receives the response message, it first extracts the response payload data from the SOAP message. Then, an IMS Connect message is built to send the response message back to the IMS Appl2. The IMS Connect message has the format, LLLLIRM<Response data>. The IRM contains the transaction code for the IMS Appl2. It also contains the XML Adapter name and the converter name if the user chooses to use the XML adapter function to transform the callout response message. The transaction code, XML adapter, and converter name are all obtained from the correlator file specified by the user during the deployment step. If the user chooses not to use the XML adapter, IMS SOAP Gateway adds LL ZZ and trancode in front of the response data.

After the response message is built, it is sent to IMS Connect using Commit Mode 0 with Send Only protocol.

When IMS Connect receives the XML response message from the IMS SOAP Gateway, the XML conversion function converts the XML response data into the IMS application data format if the user chooses the XML adapter function. Upon receiving the converted application data bytes from the converter, the XML Adapter computes the new LL value. It also sets the transaction code value (obtained from the IRM that IMS SOAP Gateway sends) within the message. The resulting response data message is then sent to IMS Appl2.

Note: If the callout response needs to go back to the initial client, a synchronous callout solution is recommended. However, a pseudo synchronous solution can be achieved with appropriate application design, for example, the initial IMS application can wait in the dependent region and continue to check for the output data (updated by a second transaction) through a database.

10.4.3 Web Services callout scenarios for Asynchronous Callout

In this section, we describe three common scenarios. All of these scenarios use the IMS Connect XML Adapter function for XML transformation and the OTMA descriptor function.

Scenario 1: Callout to one-way Web Service with no response to IMS

In this scenario, presented in Figure 10-4, the IMS application calls out to an one-way operation on the Web Service. Neither response or error is expected from the Web Service. An example of this type of processing is when an IMS application needs to invoke a Web Service to perform a task, such as callout to a printer.

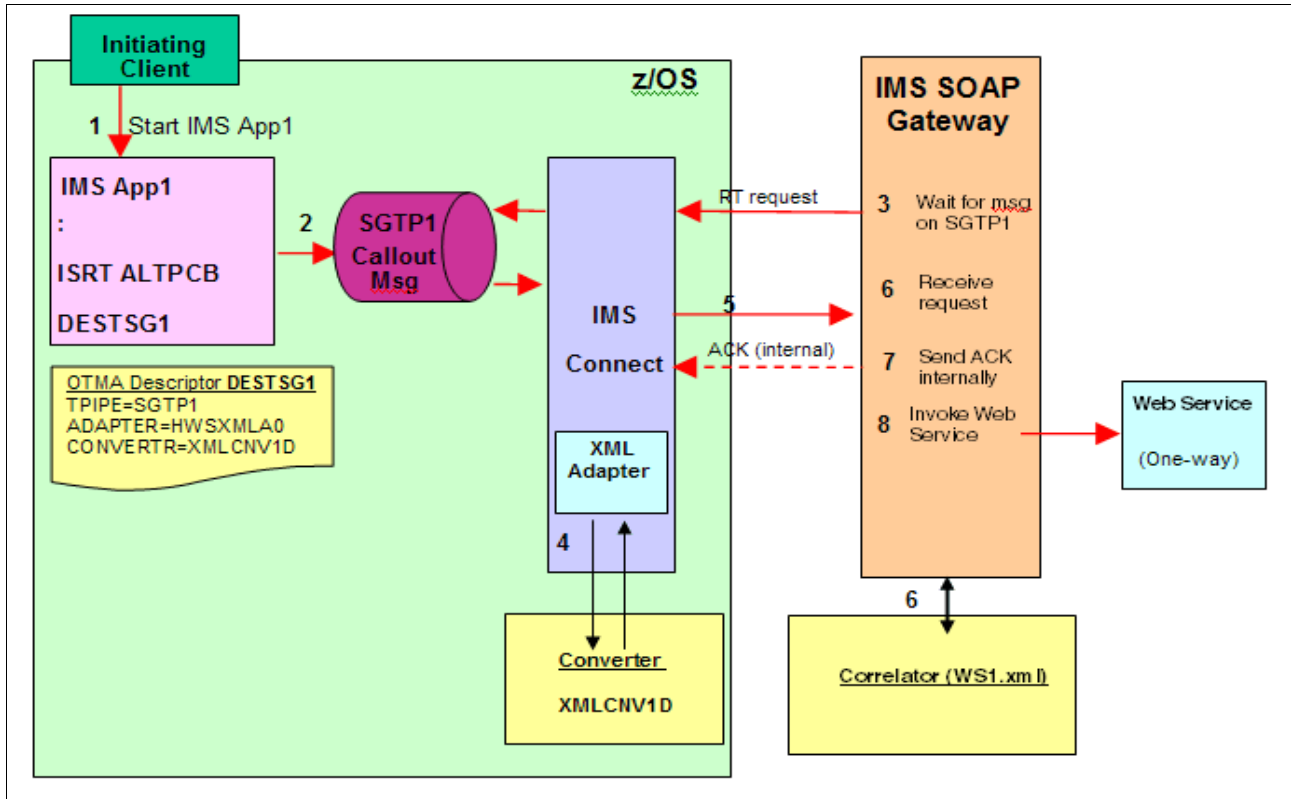


Figure 10-4 SOAP Callout to a one-way Web Service invocation with no response

The stages that are associated with this type of asynchronous callout flow are:

1. A client starts IMS App1.
2. A one-way callout message is inserted into the ALTPCB by the IMS application App1 using the destination routing descriptor DESTSG1. The descriptor contains the TMEMBER name, TPIPE name, XML Adapter name, and the associated Converter name.

As part of the destination routing descriptor processing, OTMA reads the descriptor DESTSG1, which contains a TPIPE name of SGTP1, and sends the callout message to the asynchronous hold queue SGTP1 TPIPE. The adapter name (HWSXMLA0) and converter name (XMLCNV1D) specified in the DESTSG1 descriptor are copied to the OTMA headers of the callout message. The callout message looks like Figure 10-5, after descriptor processing.

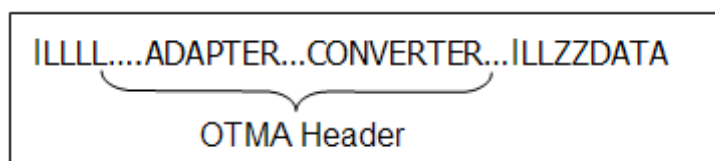


Figure 10-5 Callout message after OMTA descriptor processing

3. IMS SOAP Gateway establishes a sharable persistent TCP/IP connection with IMS Connect and sends a RESUME TPIPE using an option of 'No Auto' to retrieve messages from the SGTP1 TPIPE. The TPIPE is obtained from the connection bundle file when IMS SOAP Gateway starts up.
4. IMS Connect retrieves the existing callout message from SGTP1 as part of the RESUME TPIPE request processing for IMS SOAP Gateway. It reads the OTMA headers for the adapter name and converter name. It calls the adapter HWSXMLA0 and passes the converter name XMLCNV1D to it.

After the activity completes the XML Adapter calls the converter XMLCNV1D to perform the data transformation from COBOL application data to XML. The converter adds the service data prefix to the message so that IMS SOAP Gateway can use/read the appropriate correlator file and direct the request to the appropriate Web Service operation.

5. After the converter converts the callout message successfully, XML Adapter wraps the callout message in a <XMLAdapterOutput> tag and then IMS Connect adds a 4-byte length LLLL at the beginning of the message and sends the callout request message back to IMS SOAP Gateway. The resulting callout message appears in Figure 10-6.

```
LLLL<XMLAdapterOutput><IOGCallout><IOGServiceData><WSID>WS1</WSID>...</IOGServiceData><Req>DATA</Req></IOGCallout></XMLAdapterOutput>LLZZ*CSM OKY
```

Figure 10-6 Callout message with XML Adapter Output tag

6. After IMS SOAP Gateway receives the callout request message, it reads the LLLL portion of the message and determines the length of the message to be read. It also parses the message to retrieve the WSID from the service data prefix. Based on the WSID value "WS1", it loads the corresponding correlator file, which is WS1.xml, and retrieves the Web Service information and properties for invoking the Web Service. The request message for Web Service is built.
7. After the callout message is processed successfully and the request message for the Web Service is built, IMS SOAP Gateway internally sends a positive acknowledgement (ACK) to IMS Connect such that the callout message can be removed from the corresponding TPIPE.
8. IMS SOAP Gateway sends the one-way callout request message to the Web Service using SOAP/HTTP. The callout message using SOAP/HTTP appears in Figure 10-7.

```
<SOAP><Req>DATA</Req></SOAP>
```

Figure 10-7 Callout message with SOAP tag

Scenario 2: Callout to one-way Web Service with a response returned to IMS by invoking a second Web Service on IMS SOAP Gateway

In this scenario, presented in Figure 10-8 on page 213, IMS application App1 calls out as a one-way operation to Web Service 1 which invokes another one-way operation on Web Service 2.

The first seven steps are similar to the previous scenario in that they provide details of how a Web Service is invoked. The additional steps show the details of a Web Service invoking another pre-deployed Web Service, which calls an existing IMS application.

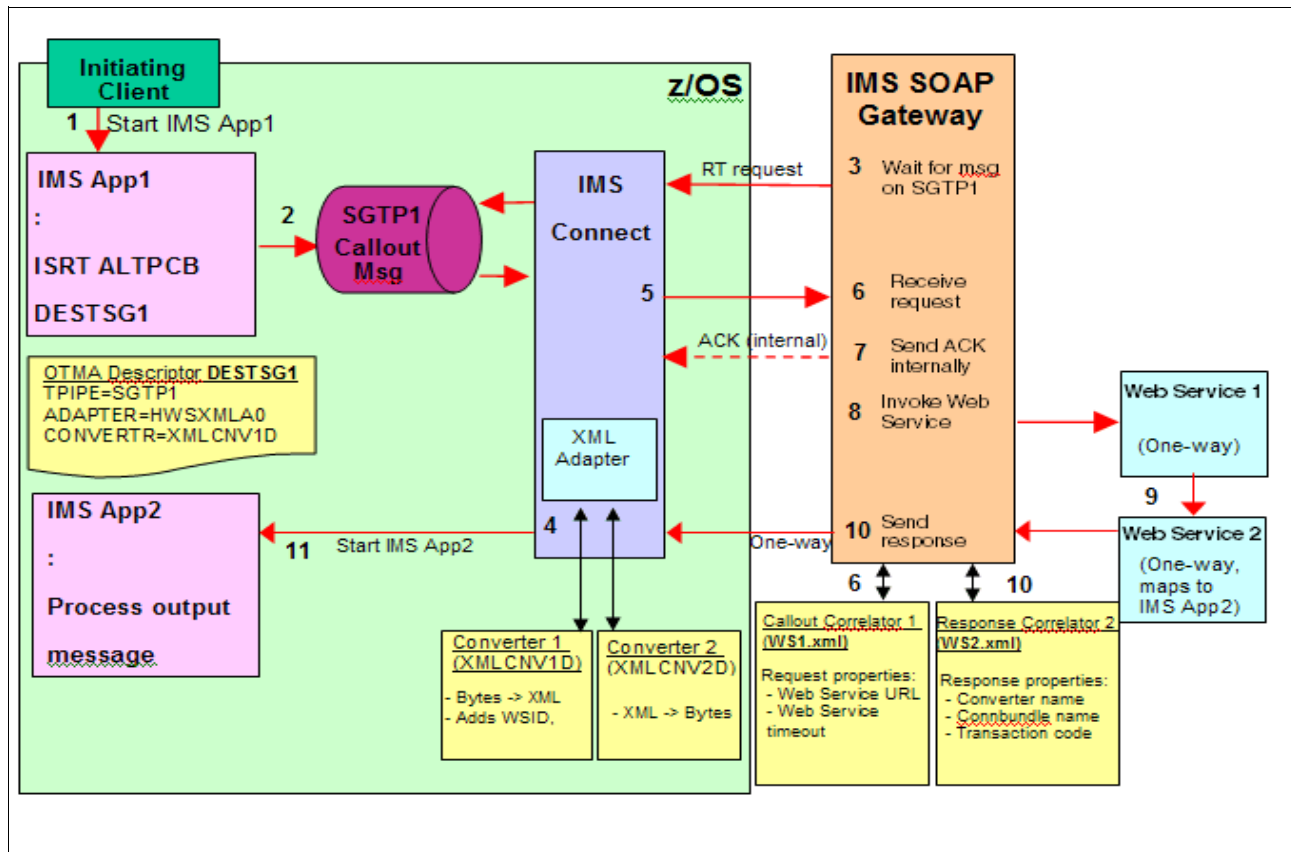


Figure 10-8 SOAP Callout to a one-way Web Service invocation with a response

1. A client starts IMS application App1.
2. A one-way callout message is inserted into the ALTPCB by the IMS application App1 using the destination routing descriptor DESTSG1. The descriptor contains the TMEMBER name, TPIPE name, XML Adapter name, and the associated Converter name.

As part of the destination routing descriptor processing, OTMA reads descriptor DESTSG1, which contains a TPIPE name of SGTP1, and sends the callout message to the asynchronous hold queue SGTP1 TPIPE. The adapter name (HWSXMLA0) and converter name (XMLCNV1D) specified in the DESTSG1 descriptor are copied to the OTMA headers of the callout message. The callout message looks like Figure 10-5 on page 211 after descriptor processing.

3. IMS SOAP Gateway establishes a sharable persistent TCP/IP connection with IMS Connect and sends a RESUME TPIPE to retrieve messages from the SGTP1 TPIPE. The TPIPE is obtained from the connection bundle file when IMS SOAP Gateway starts up.
4. IMS Connect retrieves the existing callout message from SGTP1 as part of the RESUME TPIPE request processing for IMS SOAP Gateway. It reads the OTMA headers for the adapter name and converter name. It calls the adapter HWSXMLA0 and passes the converter name XMLCNV1D to it.

The XML Adapter calls the converter XMLCNV1D to perform the data transformation from COBOL application data to XML. The converter adds the WSID and the operation name information to the message so that IMS SOAP Gateway can use/read the appropriate correlator file and direct the request to the appropriate Web Service operation.

5. After the converter converts the callout message successfully, the XML Adapter wraps the callout message in a <XMLAdapterOutput> tag and then IMS Connect adds a 4-byte

length LLLL at the beginning of the message and sends the callout request message back to IMS SOAP Gateway. The resulting callout message looks like Figure 10-6 on page 212.

6. After IMS SOAP Gateway receives the callout request message, it reads the LLLL portion of the message and determines the length of the message to be read. It also parses the message to retrieve the service data. Based on the WSID value “WS1” from the service data, it loads the corresponding correlator file, which is WS1.xml, and retrieves the Web Service information and properties for invoking the Web Service. The request message for Web Service is built.
7. After the callout message is processed successfully and the request message for the Web Service is built, IMS SOAP Gateway internally sends a positive acknowledgement (ACK) to IMS Connect such that the callout message can be removed from the corresponding TPIPE.
8. IMS SOAP Gateway sends the one-way callout request message to the Web Service 1 using SOAP/HTTP. No response is expected from the Web Service 1. The callout message using SOAP/HTTP appears as in Figure 10-7 on page 212.
9. Web Service 1 executes, generates output, and calls Web Service 2. It is the responsibility of Web Service 1 to create and transform output in such a manner that it maps to the input request of Web Service 2.
10. IMS SOAP Gateway receives the Web Service 2 request and it loads the correlator 2 (WS2.xml) to obtain the information for IMS Appl2 and sends the data to IMS Connect. In this scenario, the IMS transaction code is supplied by the Web Service 2.
11. IMS Connect receives the input message and loads converter 2 (XMLCNV2D) for XML to bytes conversion. Then, it invokes the IMS application Appl2 to process the data.

Scenario 3: Callout to a request-response Web Service

In this scenario, presented in Figure 10-9 on page 215, the IMS application calls out to a request-response operation on the Web Service, and a response is sent back to IMS SOAP Gateway, which invokes a new IMS application (IMS application App2) for processing the output.

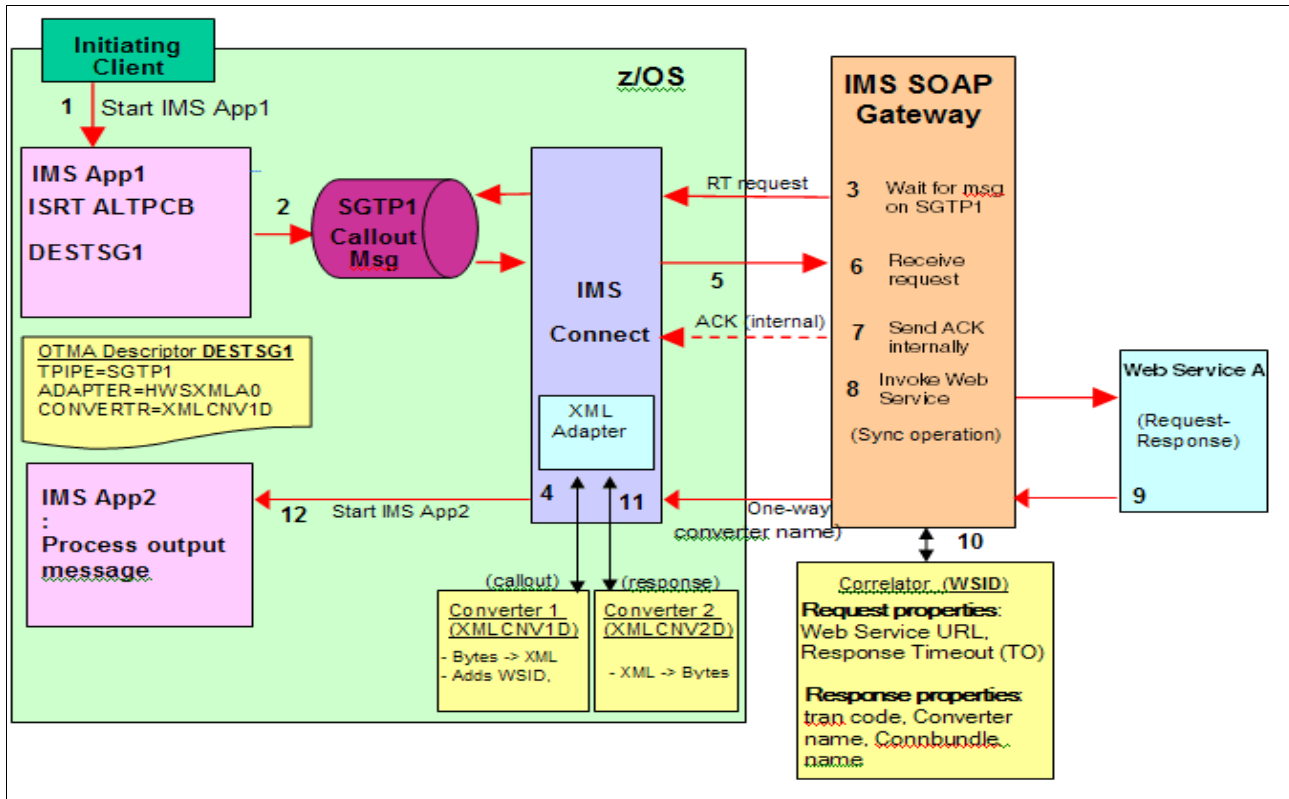


Figure 10-9 Message flow of SOAP Callout to a request-response Web Service invocation

1. A client starts IMS application App1.
2. A one-way callout message is inserted into the ALTPCB by IMS application App1 using the destination routing descriptor DESTSG1. The descriptor contains the TMEMBER name, TPIPE name, XML Adapter name, and the associated Converter name. The callout message looks like Figure 10-6 on page 212 after descriptor processing.

As part of the destination routing descriptor processing, OTMA reads the descriptor DESTSG1, that has a TPIPE name of SGTP1, and sends the callout message to the asynchronous hold queue SGTP1 TPIPE. The adapter name (HWSXMLA0) and converter name (XMLCNV1D) specified in the DESTSG1 descriptor are copied to the OTMA headers of the callout message.

3. IMS SOAP Gateway establishes a sharable persistent TCP/IP connection with IMS Connect and sends a RESUME TPIPE using an option of 'No Auto' to retrieve messages from the SGTP1 TPIPE. The TPIPE is obtained from the connection bundle file when IMS SOAP Gateway starts up.
4. IMS Connect retrieves the existing callout message from SGTP1 as part of the RESUME TPIPE request processing for IMS SOAP Gateway. It reads the OTMA headers for the adapter name and converter name. It calls the adapter HWSXMLA0 and passes the converter name XMLCNV1D to it.

The XML Adapter calls the converter XMLCNV1D to perform the data transformation from COBOL application data to XML. The converter adds the WSID and the operation name information to the message so that IMS SOAP Gateway can use/read the appropriate correlator file and direct the request to the appropriate Web Service operation.

5. After the converter converts the callout message successfully, XML Adapter wraps the callout message in <XMLAdapterOutput> tag and then IMS Connect adds a 4-byte length

LLLL at the beginning of the message and sends the callout request message back to IMS SOAP Gateway. The resulting callout message appears as in Figure 10-6 on page 212.

6. After IMS SOAP Gateway receives the callout request message, it reads the LLLL portion of the message and determines the length of the message to be read. It also parses the message to retrieve the service data prefix. Based on the WSID value “WS1” from the service data prefix, it loads the corresponding correlator file, which is WS1.xml, and retrieves the Web Service information and properties for invoking the Web Service. The request message for Web Service is built.
7. After the callout message is processed successfully and the request message for the Web Service is built, IMS SOAP Gateway internally sends a positive acknowledgement (ACK) to IMS Connect such that the callout message can be removed from the corresponding TPIPE.
8. IMS SOAP Gateway sends the request to the Web Service 1 using SOAP/HTTP and waits for the response synchronously on the same HTTP connection. The callout message using SOAP/HTTP appears as in Figure 10-7 on page 212.
9. The Web Service 1 runs and generates output response and sends back to IMS SOAP Gateway.
10. IMS SOAP Gateway adds the transaction code App2, the adapter name, HWSXMLA0; and the converter name, XMLCNV2D (all retrieved from the correlator file in step 6) to the IRM and sends the response message to IMS Connect.
11. IMS Connect receives the response message and loads the response converter XMLCNV2D for XML to bytes conversion. IMS Connect takes the transaction code value and adds it to the appropriate place (LLZZTRCDDATA) in the message.
12. IMS invokes the IMS application App2 to process the output data.

RDz can be used to import the SOAP Client WSDL, generate the correlator file, create the XML Converter routine to be called by IMS Connect to do the XML transformation for the callout message, and the COBOL copy member that represents the output message.

Considerations for Callout usage

When you use the callout function, to restart or gracefully shut down the IMS SOAP Gateway server:

1. Stop all the callout threads by using the IMS SOAP Gateway Deployment utility. Give sufficient time for the threads to stop completely. The time it takes to stop the threads is based on the callout poll interval property that is defined in the imsssoap.properties file. At a minimum, wait as long as the callout poll interval time.
2. Stop the IMS SOAP Gateway server.
3. When you are ready, start the IMS SOAP Gateway server. If the callout function is not disabled, the callout threads would start automatically.

10.5 Synchronous Callout with IMS SOAP Gateway

IMS SOAP Gateway 10.1 with the latest iFix supports synchronous callout. With this new feature your IMS applications are enabled to synchronously invoke external applications

With Synchronous Callout support, IMS applications callout synchronously and wait for the response to come back. Using this programming style, the IMS application program can initiate direct communication with other external application programs and receive the response in the same IMS transaction instance.

10.5.1 Detailing Synchronous Callout with IMS SOAP Gateway

Figure 10-10 is an overview of IMS Synchronous Callout support.

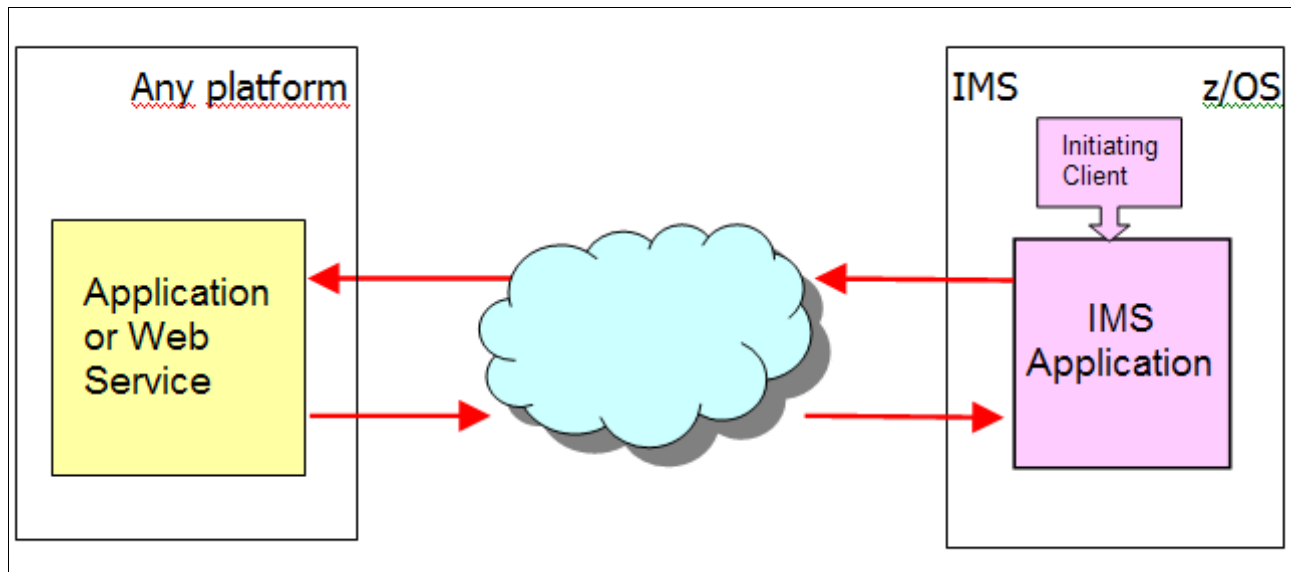


Figure 10-10 IMS Synchronous Callout support

The IMS Synchronous Callout support enhancements are:

- ▶ A new DL/I call, the ICAL SENDRECV call, for an IMS application to send out a synchronous callout request and receive the response back. The ICAL synchronous call function introduces a new IMS application programming style. The traditional IMS application programming style is an asynchronous programming model supported by the IMS message queue.
- ▶ The ICAL synchronous call function is only supported using the AIBTDLI interface in IMS TM runtime environments for IFP, MPP, BMP, JMP, and JBP regions. ICAL is used to support IMS Synchronous Callout.
- ▶ Synchronous invocation to Web Services.
- ▶ Communication topology to enable an external application to process the callout request and to return the response to the same IMS application program instance. The IMS-dependent region is in a wait state when waiting for the response.
- ▶ Time-out capability to allow the IMS application to specify how long it can wait for the response message to come back, which can help to terminate the callout request and free up the dependent region if the external service does not respond in a timely manner.
- ▶ Enhanced OTMA Resume TPIPE and Send-Only message processing functions for external application to pull synchronous callout request messages and return the response back.
- ▶ Relief of the 32K segmentation limitation for synchronous callout messages because the IMS message queue is not used. The maximum XML message size for both the request and response for synchronous callout messages is 32 MB for COBOL applications and 16 MB for PL/I applications, which enables IMS applications to send out and receive large messages.
- ▶ Concurrent processing with a correlation mechanism to allow callout requests to be sent out and response messages to be received in different connections and threads.
- ▶ Updated IMS commands to view synchronous callout request status.

Figure 10-11 shows an overview of the synchronous callout flow.

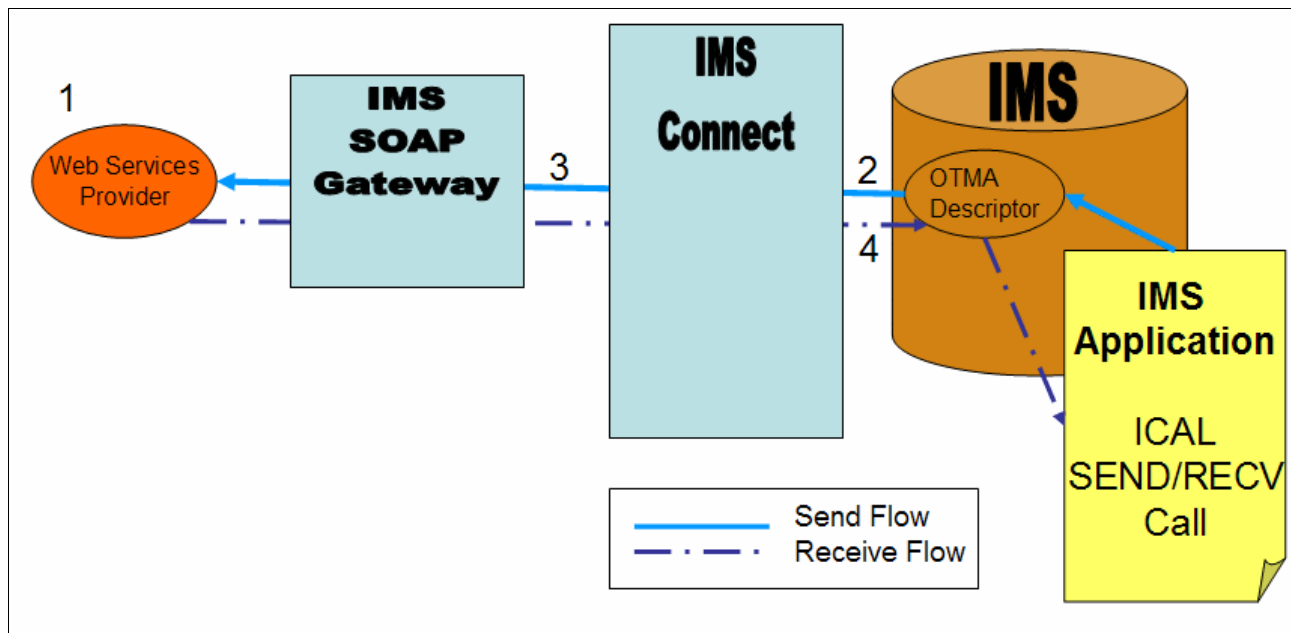


Figure 10-11 Overview of IMS Synchronous Callout flow

The flow of the synchronous callout flow is:

1. The external application or server does not listen for callout messages because of the use of the IMS OTMA Resume TPIPE function.
2. The IMS application program issues an ICAL SENDRECV synchronous callout request through an OTMA descriptor name, which defines the destination for the call.
3. The OTMA descriptor allows IMS to route the callout request through IMS Connect to the outbound destination, for example, a EJB/MDB running in WebSphere Application Server using the IMS TM Resource Adapter, a RYO IMS Connect TCP/IP application, or a Web Service provider through IMS SOAP Gateway.
4. After the callout request is processed, the response data is returned back to the same IMS transaction instance.

In summary, after the synchronous callout request message is received, IMS SOAP Gateway processes the request, finds the Web Service identifier in the callout request message and matches it with the deployed Web Services to locate and invoke the desired Web Service. The invoked Web Service, after processing, sends the callout response message back to the IMS application that is waiting synchronously for a response.

Invoking the Web Service operation

For Synchronous Callout, only one type of Web Service operation is supported. It is the request-response invocation where a response message is returned synchronously back to the caller. IMS SOAP Gateway waits for the response data to come back synchronously from the same SOAP/HTTP connection.

The invocation of the Web Service is performed by a worker thread. Any errors that occur in the worker thread during the invocation of the Web Service is sent back with an error response to IMS Connect which, in-turn, passes the error to the waiting IMS application.

10.5.2 Synchronous Callout User Scenarios

In this section, we describe the various topologies where you can use synchronous Callout.

Scenario 1: Multiple IMS applications

Figure 10-12 illustrates that multiple IMS applications can make callout requests at the same time. In this example, each IMS application uses a different descriptor for the target callout destination.

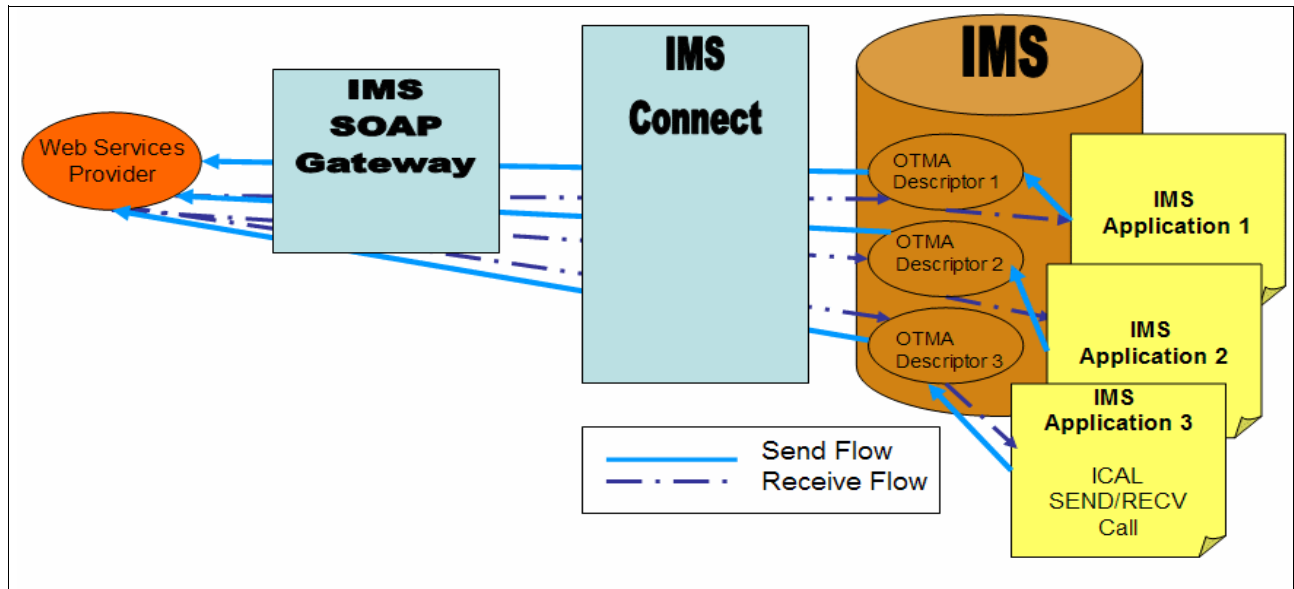


Figure 10-12 Callout flow with multiple IMS applications

Scenario 2: Multiple IMS Connects

Figure 10-13 on page 220 represents the scenario that different IMS Connect instances are used to send the request and to receive the response messages. The correlation token sent inside the request and response message allows IMS to ensure that the response message goes to the same IMS transaction instance that initiates the synchronous callout request.

This scenario is common if Sysplex Distributor is configured and used in front of a number of IMS Connects. All the IMS Connects would share a single external IP address that is managed by the Sysplex Distributor. When the response is returned, the sysplex distributor might route the response to any of the IMS Connect if the external server uses a different connection to return the callout response message.

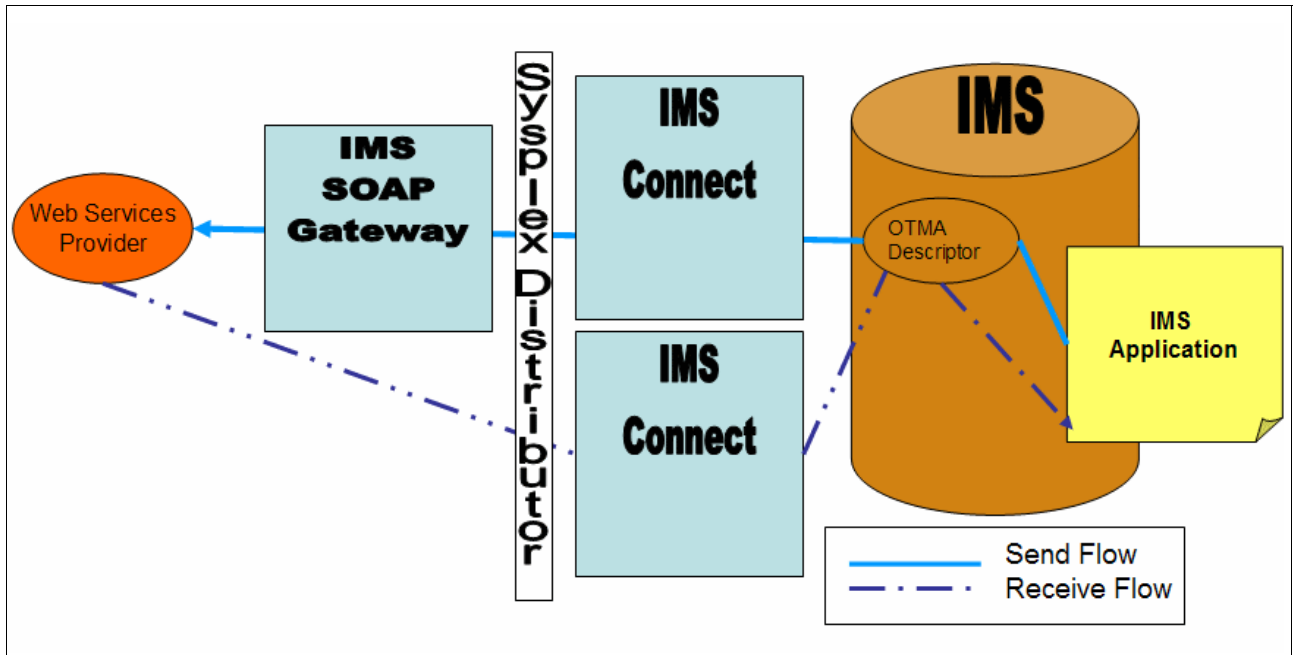


Figure 10-13 Callout flow with multiple IMS Connects

Scenario 3: IMS Shared Queues

Figure 10-14 illustrates that the Synchronous Callout support can be used in a shared queue environment if the IMS is both the front-end and back-end system.

The Send Only response must be routed back to the same IMS that initiated the request; otherwise, the response message is rejected.

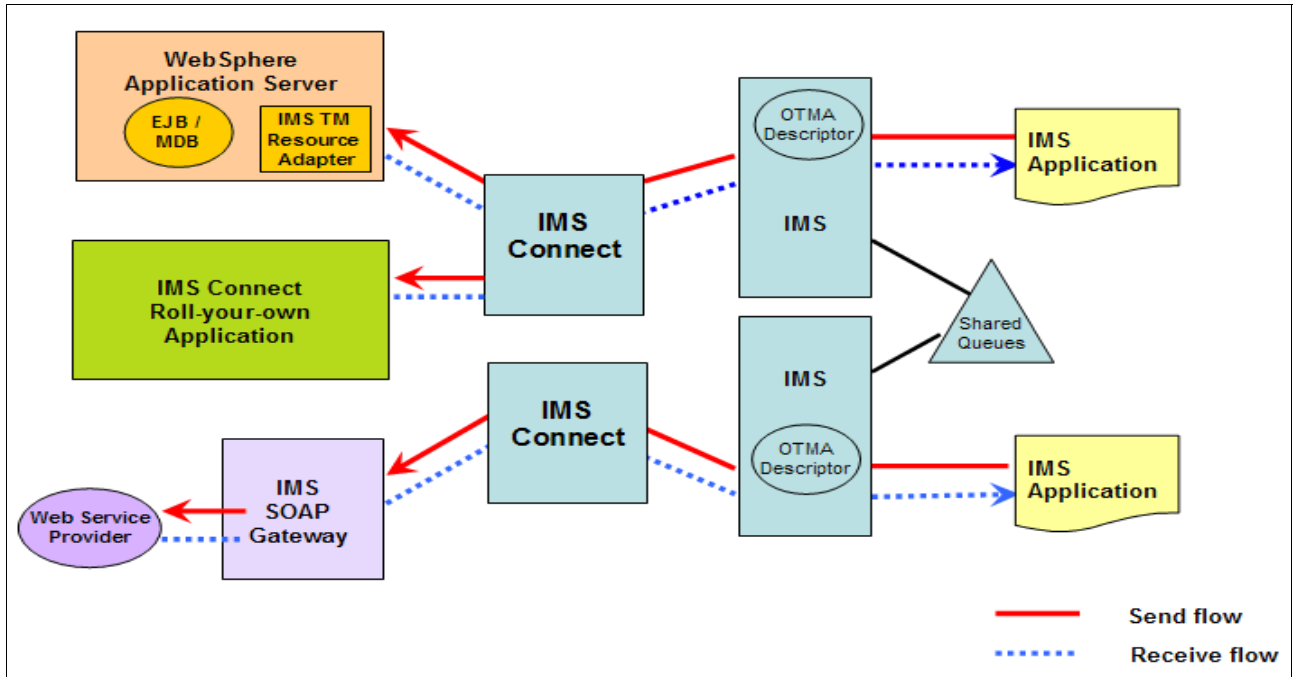


Figure 10-14 Callout flow with IMS Shared Queues

Scenario 4: Multiple Servers for High Availability

Figure 10-15 illustrates that multiple servers can be configured to achieve high availability and redundancy.

Server 1 can be a WebSphere Application Server, IMS SOAP Gateway, or RYO application. Server 2 can be connected to the same IMS Connect as Server 1 or a different IMS Connect instance. If IMS Connect attempts to deliver the callout request message to Server 1 and fails, IMS Connect sends the message back to OTMA, which allows the message to remain on the TPIPE and be retrieved by Server 2 to complete the processing.

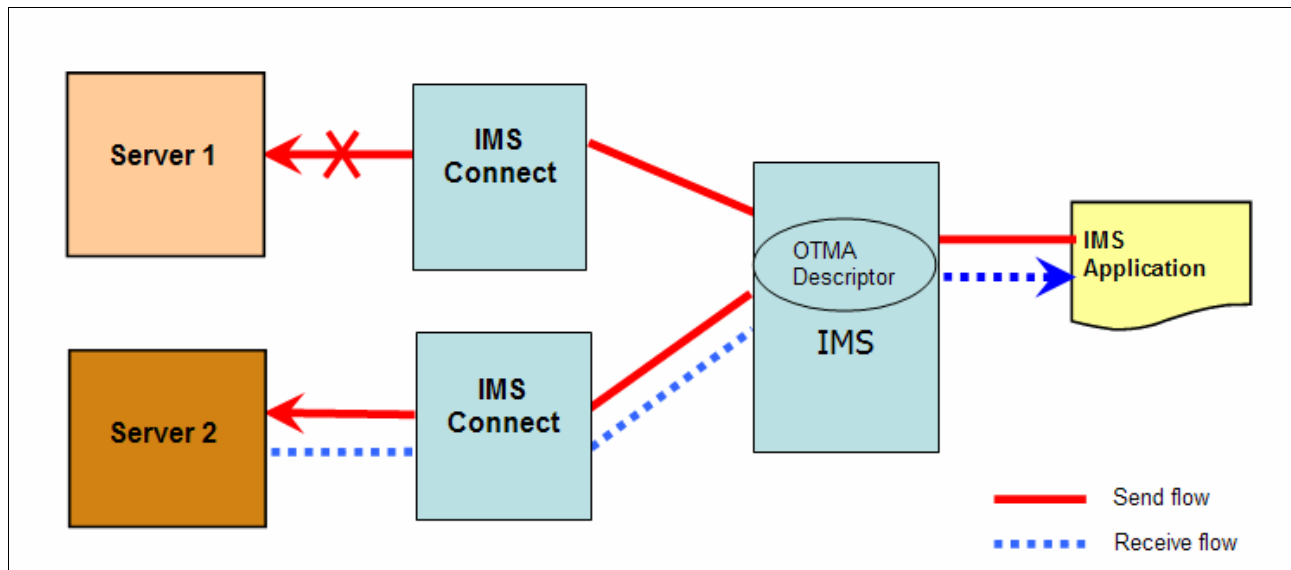


Figure 10-15 Callout flow with multiple servers

Requirements and restrictions for synchronous callout

In this section, we provide the requirements and restrictions for synchronous callout.

The requirements for synchronous callout are:

- ▶ IMS SOAP Gateway Version 10.1 with the latest iFix
- ▶ Rational Developer for System z Version 7.1.1.3 or later

The restrictions for synchronous callout are:

- ▶ Two-phase commit is not supported.
- ▶ Shared queue environments with different front end and back end IMS systems is not supported. The response message of a synchronous callout request must flow back to the originating IMS to be processed correctly.
- ▶ BMP or JBP applications running in a DBCTL environment are not supported.
- ▶ The callout processing inside the IMS application must be single-threaded, that is, the IMS application can only issue one synchronous callout call at a time. It must wait for the response (or time-out) before it can issue the next callout request.
- ▶ The maximum XML message size for both the request and response for synchronous callout messages, is 32 MB for COBOL applications and 16 MB for PL/I applications.

10.6 Multi-segment support

IMS SOAP Gateway 10.2 introduces multi-segment support. The significance of this support is that IMS multi-segment applications can now participate in SOA using IMS SOAP Gateway.

This support is provided for the XML adapter scenario in IMS Connect and IMS SOAP Gateway, with tooling support from Rational Developer for System z Version 7.5. Multi-segment support is provided only for the “provider” (inbound – both request / response) scenario and not for the “consumer” (callout) scenario.

As with current single segment handling, the bulk of the multi-segment handling is performed by the RD/z generated XML converters, which are deployed into IMS Connect. The IMS Connect XML Adapter function utilizes the XML converters for the XML-to-bytes conversion on the way in and bytes-to-XML conversion on the way out of IMS. This approach still applies for multi-segment support. IMS SOAP Gateway utilizes the RD/z tooling to support the generation of Web Services artifacts for multi-segment message processing programs (MPPs).

Multi-segment handling in SOAP Gateway is achieved in two steps:

1. Artifact generation using Rational Developer for System z (RD/z) Version 7.5.
2. Runtime handling in IMS Connect by the XML Adapter and the generated XML converters.

The pre-requisites for multi-segment support are:

- ▶ IMS Connect 10 SPE APAR PK69366
- ▶ IMS SOAP Gateway 10.2
- ▶ Tooling support: Rational Developer for System z 7.5

10.7 Security enhancements

SSL and HTTPS security allow data to be transferred in a secure manner between the Web Service and IMS SOAP Gateway and between IMS SOAP Gateway and IMS Connect.

IMS SOAP Gateway processes a SOAP request by stripping off the SOAP headers and passing the XML payload to IMS Connect. This can be sent in plain format or over a SSL connection to protect the message.

The security properties between IMS SOAP Gateway, IMS Connect, and IMS are specified in the connection bundle. The security properties that must be specified and that are passed to IMS Connect are: User Id, Password, and Groupname.

In addition, there are SSL parameters that are required for the IMS SOAP Gateway to communicate with the IMS Connect through TCP/IP SSL Sockets. These parameters are listed in 10.1, “IMS SOAP Gateway implementation overview” on page 204. To specify this data you must open the Connection Bundle file, which can be accessed through the IMS SOAP Gateway Deployment utility.

If RACF=Y is specified in the HWSCFGxx member, IMS Connect issues a RACF call to verify these parameters. If the call is not successful, an error message is returned to IMS SOAP Gateway. If the call is successful, the UserID and Group is passed to OTMA.

The IMSLSECX exit is also invoked if it exists, but this exit does not verify the password. Figure 10-16 on page 223 maps out the message flow under security controls.

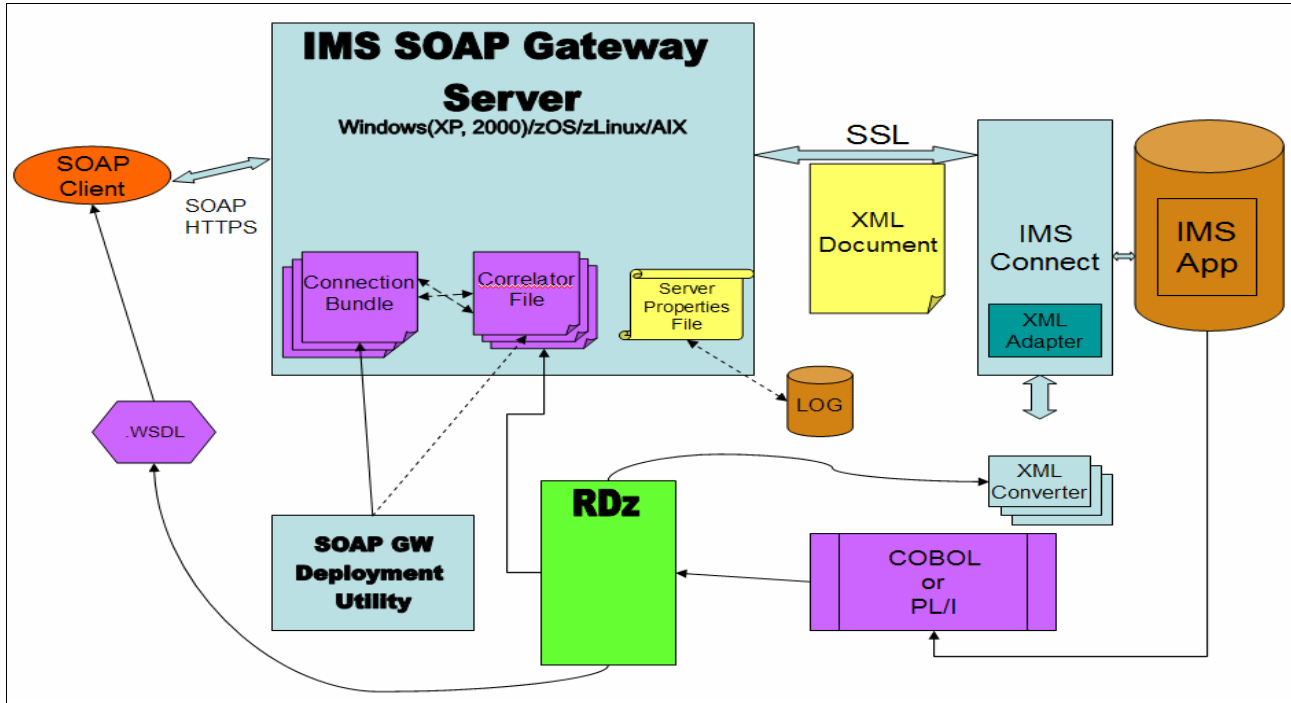


Figure 10-16 SSL/HTTP message flow

10.8 IMS SOAP Gateway features and compatibilities

Table 10-1 maps what features of IMS SOAP Gateway are compatible to which tools and versions of IMS and IMS Connect.

Table 10-1 Summary of IMS SOAP Gateway features and compatibilities

Feature	Minimum SOAP Gateway version	Minimum IMS version	Minimum IMS Connect version	Minimum Rational Tooling version
Accessibility Enablement	v 9.2	v9	v9	7.0
IMS Connect XML Adapter Support	v9.2	v9	v9	7.0
Security Enhancements: SSL/HTTPS	v 9.2	v9	v9	7.0
Platform Support: z/OS and zLinux	zLinux: v9.2 zOS: v10	v9 v10	v9 v10	v7.0 v7.1.1
Asynchronous Callout	v10.1	v10	v10	7.1.1
Synchronous Callout	v10.1 +iFIX2	v10	v10	7.1.1.3
Multiple Segment	v10.2	v10	v10	v 7.5

Refer to the following Web site for the current SOAP Gateway compatibility table:

<http://www-01.ibm.com/software/data/ims/soap/>

Multiple segment support in IMS SOAP Gateway is announced in the IMS Version 11 QPP (Quality Partnership Program) announcement number 208-258 found in URL:

http://www.ibm.com/common/ssi/index.wss?DocURL=http://www.ibm.com/common/ssi/rep_ca/8/897/ENUS208-258/index.html&InfoType=AN&InfoSubType=CA&InfoDesc=Announcement+Letters&panelurl=index.wss%3Fbuttonpressed%3DNAV002PT090&paneltext=Announcement+letter+search

This concludes our discussion about SOAP Gateway.



The IMS TM Resource Adapter

The IBM IMS TM Resource Adapter (previously known as IMS Connector for Java (IC4J)) is used by Java applications, Java Platform, Enterprise Edition (Java EE) applications, or Web Services to access IMS transactions that are running on host IMS systems.

It is also used by IMS applications that run in IMS dependent regions to make asynchronous and synchronous (supported through IMS Version 10 maintenance and Version 11) callout requests to external Web Services.

In addition, the IMS TM Resource Adapter implements the J2C Common Client Interface (CCI), which is a programming interface that allows your application to communicate with the IMS Transaction Manager.

The IMS TM Resource Adapter is used with a Java EE server, such as IBM WebSphere Application Server, when Java applications (Java EE artifacts in the WebSphere Application Server) access an IMS transaction that is running on a host IMS system. The IMS TM Resource Adapter also enables an IMS application to act as a client to invoke applications in a Java EE server.

Although the IMS TM Resource Adapter is intended for use primarily by Java applications or Web Services that submit transactions to IMS, the IMS TM Resource Adapter can also be used by services that submit IMS commands to IMS.

To have the full benefit of the adapter code, the adapter must be installed in the WebSphere Application Server and used from Java EE artifacts, but RYO Java client programs can also utilize the code.

11.1 Key features of the IMS TM Resource Adapter

Support is based on Sun's Java EE Connector Architecture (JCA) 1.5 for the IMS TM Resource Adapter version levels of 10, 11 and 9.1.0.2.x. Level JCA 1.0 is associated with versions V9.1.0.1.x and V2.2.x. TMRA provides the following support:

- ▶ Supports development of applications that can submit transactions to IMS Transaction Manager through IMS Connect.
- ▶ Provides tooling support for development of Java EE applications, Web Services, and business processes that access IMS transactions in various Rational and WebSphere integrated development environments.
- ▶ Provides a JCA Resource Adapter (RAR) for deployment to the WebSphere Application Server and WebSphere Process Server (WPS) runtime environment on many platforms including z/OS and Linux on System z.
- ▶ Supports connection pooling and reuse.
- ▶ Supports global transaction processing and two-phase commit.
- ▶ Supports component-managed and container-managed security, including support for run-as thread identity.
- ▶ Supports SSL communication between the IMS TM Resource Adapter and IMS Connect, including support for SSL null encryption.
- ▶ Supports most types of interactions with IMS application programs including the retrieval of asynchronous output messages.
- ▶ Supports for IMS applications to invoke Java EE applications asynchronously, and synchronously in callout mode.

IMS TMRA components

There are two key components that make up TMRA:

- ▶ A runtime component that needs to be deployed on the WebSphere Application Server. A deployed TM resource adapter in WebSphere Application Server fulfills all rules of the JCA 1.5 contract.
- ▶ A development component that lets you create your application by using an integrated development environment (IDE), such as IBM Rational Application Developer for WebSphere Software, IBM Rational Software Architect, IBM WebSphere Integration Developer, and IBM WebSphere Transformation Extender, as part of the optional J2C feature.

The J2C wizards in these IDEs enables you to create IMS connector code to be used from Enterprise JavaBeans (EJB) components, Web Services, or from standalone programs.

Supported platforms

The IMS TMRA runtime component supports WebSphere Application Server on the following platforms:

- ▶ AIX®
- ▶ HP-UX
- ▶ Linux
- ▶ Linux for System z
- ▶ Solaris™
- ▶ Windows
- ▶ z/OS and OS/390

11.1.1 Features introduced in Version 10.2

IMS TM Resource Adapter Version 10.2 provides the following new features:

- ▶ Support for the invocation of IMS applications by using complex data types from distributed platforms

With the WebSphere Transformation Extender Design Studio, you can build applications that send complex data formats to and from IMS applications. WebSphere Transformation Extender is a transaction-oriented, data integration solution that automates the transformation of high-volume, complex transactions across the enterprise. You can use its Type Designer to generate a type tree from your COBOL copybook and then specify rules for transforming and routing the data in the Map Designer.

- ▶ Support for reestablishing any stale connection in a connection pool when the connection encounters a communication failure. Using this enhancement you can recycle IMS Connect during system maintenance without resubmitting any IMS TM Resource Adapter interactions from the client application.
- ▶ IMS MFS SOA support

IMS MFS SOA support is integrated with the Enterprise Metadata Discovery (EMD) framework in Rational Application Developer Version 7.5 or later to transform existing MFS-based IMS applications into J2C Java beans and J2C Java Data Binding classes. After you create a J2C Java bean, you can create Java EE artifacts, such as JSPs, EJBs, or Web Services, to deploy the generated J2C Java bean.

IMS MFS SOA support is supported in the following WebSphere Application Server V6.1 runtime environments:

- Windows
- z/OS

- ▶ Rerouting undeliverable output messages to a specified destination for commit mode 0, SYNC_SEND interactions on sharable persistent sockets

The reroute function allows undeliverable output messages to be rerouted to a specified destination. Previously, the reroute function was supported for only commit mode 0 (CM0), SYNC_SEND_RECEIVE interactions on sharable persistent sockets. This enhancement supports the reroute function on CM0, SYNC_SEND interactions on sharable persistent sockets.

11.1.2 Features introduced in Version 11

IMS TM Resource Adapter Version 11 provides the following new features:

- ▶ Support for WebSphere Transformation Extender:

WebSphere Transformation Extender (WTX) is the new name for the WebSphere DataStage® TX (formerly Ascential® DataStage TX) product. WTX performs transformation and routing of data from source systems to target systems in batch and real-time environments. The sources can include files, relational databases, MOMs (message-oriented middleware), packaged applications, or other external sources. After retrieving the data from its sources, the WebSphere Transformation Extender product transforms it and routes it to any number of targets where it is needed, providing the appropriate content and format for each target system. Figure 11-1 on page 228 positions WTX with respect to IMS TMRA.

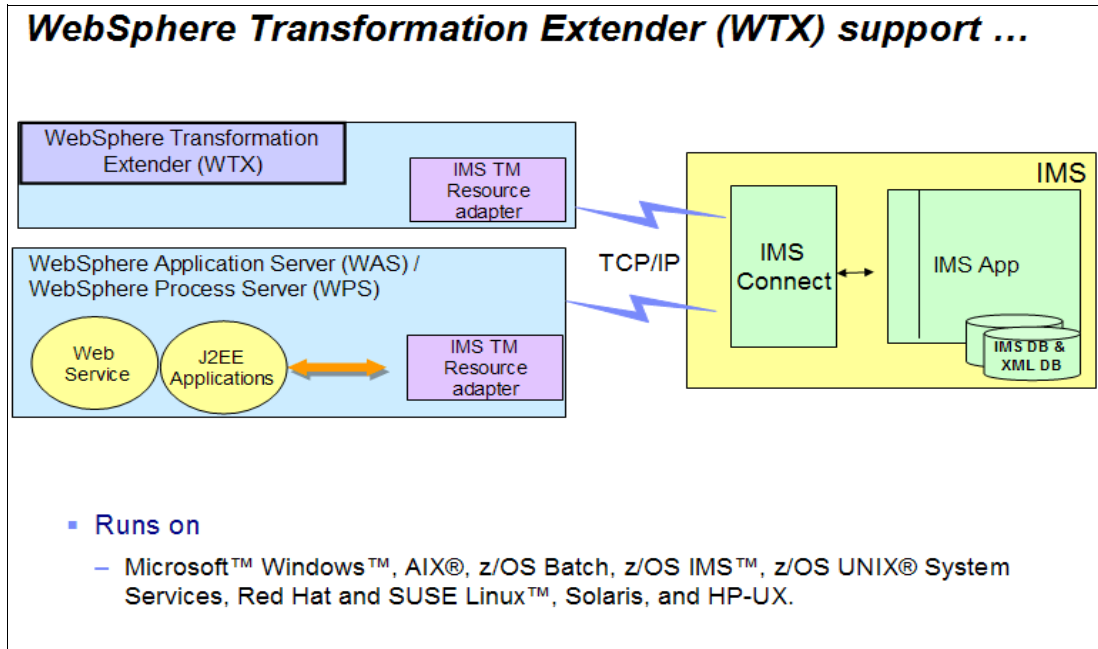


Figure 11-1 WTX support

The software requirements for WTX are:

- IMS and IMS Connect Version 10 or later
- IMS TM Resource Adapter Version 10.2 (APAR PK64663 provides this support for IMS TMRA V10.1.1)
- WTX 8.2.0.2

► **IMS TMRA socket reconnect support**

Socket reconnect support provides a retry and reconnect capability to re-establish a connection when encountering communication failures in sending / receiving and request / response through IMS Connect from a WebSphere Application Server, without user intervention. This retry function is implemented for Sync Level Confirm interactions to avoid duplicate interactions.

During system maintenance previously, customers would recycle IMS Connect, which caused WebSphere Application Server applications to fail due to persistent connections and connection pooling. This solution avoids unnecessary errors and quickly re-establishes connections, thus enhancing IMS availability and operational characteristics to allow these applications to seamlessly recover from connection errors.

► **IMS TMRA send only reroute support**

Send only reroute support reroutes current Sync-Send (Send-Only) interactions for CM0. With the reroute flag set to True and the reroute TPIPE name specified, any IOPCB output message resulting from the Send-Only transaction is queued to the reroute TPIPE.

- ▶ Message Format Services (MFS) Business Process Execution Language (BPEL) support
MFS BPEL support for the WebSphere Integration Developer assists integration developers with business-to-business transformation. This support creates Web Services/EJB/JSPs out of existing MFS-based IMS applications, and provides support for Service Component Architecture (SCA), Enterprise Metadata Discovery (EMD) 1.1, and BPEL. It includes a GUI wizard that allows you to parse the MFS source files and generate the service definition and artifacts. The generated application can be deployed to run on WebSphere Process Server for business orchestration and choreography.
- ▶ Synchronous Callout support
Synchronous callout (also available through an IMS Version 10 SPE) for IMS TMRA manages the correlation of a callout request. This enables IMS applications to invoke external applications and synchronously receive a response in the same IMS transaction instance. External applications and servers include Java EE applications (EJB/MDB).

11.2 Installing the IMS TM Resource Adapter

The latest version of the TM Resource Adapter is at:

<http://www.ibm.com/software/data/ims/ims/components/tm-resource-adapter.html>

Restriction: Synchronous callout support requires IMS TM Resource Adapter 11, 10.3

Documentation is available from *IMS TM Resource Adapter User's Guide and Reference*, SC19-1211-02.

11.3 Call-in request support

Support for external clients to request services from IMS was available for many years. This functionality can be requested from all Java EE artifacts, Servlets, EJBs, Business Processes, and Mediation Modules. The non conversational transaction type is the most simple and most frequently used. Conversational transactional flow is also supported, but does not always fit well in SOA scenarios.

When a Java application submits a transaction request to IMS through IMS TMRA, IMS Connect sends the transaction request to OTMA by using cross-system coupling facility (XCF) communications, and the transaction runs in an IMS dependent region. The response is returned to the Java application using the same path.

If access is engaged through a deployed TM Resource Adapter in the WebSphere Application Server Java EE container, the Common Client Interface (CCI) contract is honored. Figure 11-2 on page 230 shows the container component contract with respect to the other components within the container.

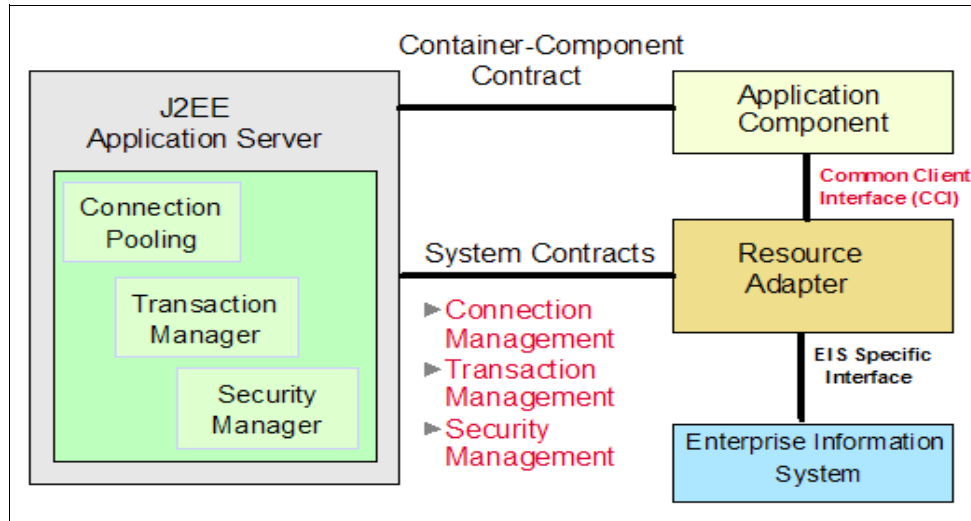


Figure 11-2 Common Client Interface contract

The CCI contract is responsible for three processing components:

- ▶ Connection management (including connection pooling). This implies that to establish the connection with IMS, a “Managed Connection Factory” is used.
- ▶ Security management through the passing of UserIDs.
- ▶ Transaction management through the control over processing units of work.

The interaction with IMS Connect and IMS is based on four Java objects:

- ▶ Connection

This object represents the connection. It is obtained from the *Connection Factory* that resides in a WebSphere Application Server. Figure 11-3 on page 231 presents the properties that can be defined in the Connection Factory

J2C connection factories

[J2C connection factories](#) > [CFIMSLN](#) > Custom properties

Use this page to specify custom properties that your enterprise information system (EIS) requires for the resource providers and resource factories that you configure. For example, most database vendors require additional custom properties for data sources that access the database.

Preferences

Name	Value	Description	Required
SSLKeyStoreName		Name (full path) of SSL keystore for client certificates/private keys	false
SSLKeyStorePassword		Password of SSL keystore for client certificates/private keys	false
UserName		Default name of the user to be authorized	false
HostName	wtsc67.itso.ibm.com	TCP/IP host name of the target IMS Connect	false
SSLTrustStorePassword		Password of SSL keystore for trusted certificates	false
SSLTrustStoreName		Name (full path) of SSL keystore for trusted certificates	false
IMSConnectName		Name of the target IMS Connect - for Local Option only	false
Password		Default password of the user	false
TraceLevel	1	Level of information to be traced.	false
DataStoreName	IMSL	Name of the target IMS datastore	false
SSLEncryptionType	Weak	The type of cipher suite to be used for encryption	false
SSLEnabled	FALSE	Indicates if SSL is enabled for this connection factory	false
GroupName		Default name of the IMS group of the user	false
PortNumber	9999	Target TCP/IP port number of IMS Connect	false
CMODedicated	FALSE	Indicates if sockets are dedicated to specific CMO clients.	false
Total 15			

Figure 11-3 Custom properties of IMS Connection Factory

Define the factory with the WebSphere Application Server Administration Console, and label it with a JNDIName for “look-up”.

Note: The factory can also be instantiated at runtime if no container (WebSphere Application Server) is available, which is also true for Java Batch, stand alone programs.

In that case, the factory must be instantiated at runtime.

► ConnectionSpec

This object provides some additional connection properties for the connection, mainly related to security and identification. See Figure 11-4 on page 232 for details.

► Interaction

This object is the coordinator of the interaction executed with IMS. Its execute method triggers the interaction.

► InteractionSpec

This object specifies the details of the interaction. See Figure 11-4 on page 232 for details.

Property Name - Type	Variable Name
<input type="checkbox"/> Connection Spec	
<input type="checkbox"/> password - String	myPassword
<input type="checkbox"/> groupName - String	myGroupName
<input type="checkbox"/> userName - String	myUserName
<input type="checkbox"/> clientID - String	myClientID
<input type="checkbox"/> Interaction Spec	
<input type="checkbox"/> purgeAsyncOutput - boolean	myPurgeAsyncOutput
<input type="checkbox"/> imsRequestType - int	myImsRequestType
<input type="checkbox"/> socketTimeout - int	mySocketTimeout
<input type="checkbox"/> mapName - String	myMapName
<input type="checkbox"/> asyncOutputAvailable - boolean	myAsyncOutputAvailable
<input type="checkbox"/> syncLevel - int	mySyncLevel
<input type="checkbox"/> useConvID - boolean	myUseConvID
<input type="checkbox"/> ltermName - String	myLtermName
<input type="checkbox"/> interactionVerb - int	myInteractionVerb
<input type="checkbox"/> convEnded - boolean	myConvEnded
<input type="checkbox"/> commitMode - int	myCommitMode
<input type="checkbox"/> convID - String	myConvID
<input type="checkbox"/> altClientID - String	myAltClientID
<input type="checkbox"/> reRoute - boolean	myReRoute
<input type="checkbox"/> reRouteName - String	myReRouteName
<input type="checkbox"/> executionTimeout - int	myExecutionTimeout

Figure 11-4 ConnectionSpec and InteractionSpec

The InteractionVerb is probably the most important parameter, which indicates what action IMS Connect must take. We remind you that the TM Resource client is always the contacting partner, even if IMS must send output.

The available verbs are:

- ▶ SYNC_SEND (value 0)
 - Sends a request to IMS when a response is not expected, in another words, perform a send only interaction.
- ▶ SYNC_SEND_RECEIVE (value 1)
 - Used for the single iteration of a non-conversational IMS transaction and for each iteration of a conversational IMS transaction.

Note: SYNC_RECEIVE (value 2) is currently not supported by IMS Connector for Java.

- ▶ SYNC_END_CONVERSATION (value 3)
 - Forces the end of an IMS conversational transaction.
- ▶ SYNC_RECEIVE_ASYNCOUTPUT (value 4)
 - Retrieves asynchronous output messages. With this type of interaction, the Java client can only receive a single message. If there are no messages in the IMS OTMA Asynchronous Queue for the clientID when the request is made, no further attempts are made to retrieve the message. No message is returned and a time-out occurs after the length of time specified in the executionTimeout property of the SYNC_RECEIVE_ASYNCOUTPUT interaction.

- ▶ `SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_NOWAIT` (value 5)
Retrieves asynchronous output messages. With this type of interaction, the Java client can only receive a single message. If there are no messages in the IMS OTMA Asynchronous Queue for the clientID when the request is made, no further attempts are made to retrieve the message. No message is returned and a time-out occurs after the length of time that is specified in the `executionTimeout` property of the `SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_NOWAIT` interaction.
- ▶ `SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT` (value 6)
Retrieves asynchronous output messages. With this type of interaction, the Java client can only receive a single message. If there are no messages in the IMS OTMA Asynchronous Queue for the clientID when the request is made, IMS Connect waits for OTMA to return a message. IMS Connect waits the length of time specified in the `executionTimeout` property of the `SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT` interaction before returning an exception.

The Generated Client ID function: In IMS Version 11, the TM Resource Adapter is enhanced to take advantage of a new function in IMS Connect: the Generated Client ID function, which ensures the uniqueness of each socket. This enhancement eliminates the requirement for IMS TM Resource Adapter to use different IMS Connect ports for instances of distributed WebSphere Application Server. IMS Connect can use the Generated Client ID function to ensure the uniqueness of each socket and allow all instances of WebSphere Application Server to specify the same IMS Connect TCP/IP port.

11.4 Callout request support

Although we call this pattern “*callout*”, in reality it starts with a “*callin*” invitation (`SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT`) from the TM Resource Adapter partner, a Java EE application.

Asynchronous callout is already available with IMS Version 10. IMS Version 11 and a SPE on IMS Version 10 (APAR PK74168) add the synchronous capability. Synchronous callout support requires the latest fix pack on WebSphere Application Server v6.1.

At least two data flows exist in a callout pattern with the WebSphere Application Server. A third inbound data flow to TMRA is required if IMS expects a response:

1. `SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT`
2. Receive callout request
3. `SYNC_SEND` (send response) optional

Figure 11-5 on page 234 illustrates this callout data flow.

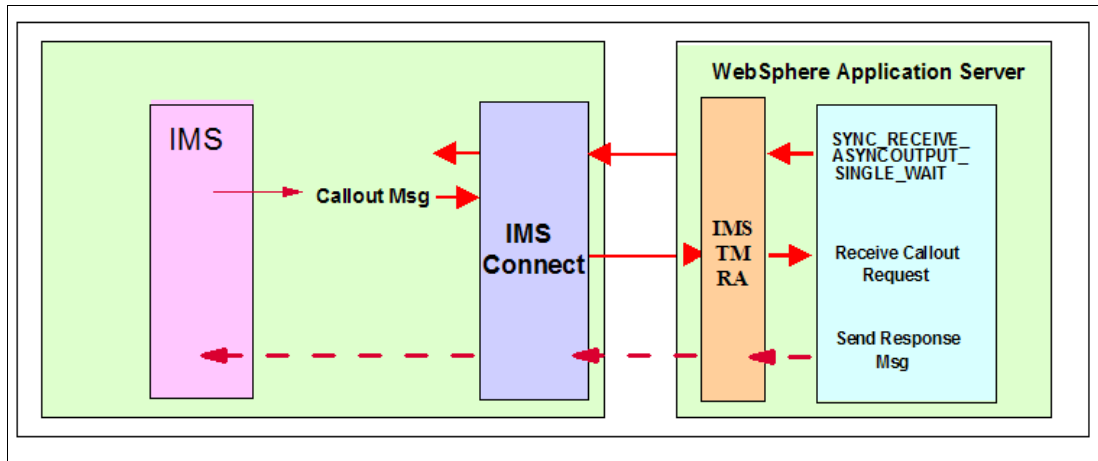


Figure 11-5 Callout data flow

OTMA destination routing descriptors are used by customers who implement this callout function or who implement message switches from OTMA to non-OTMA destinations and do not want to code the OTMA routing exits (DFSYPX0 and DFSYDRU0). If an OTMA descriptor is used with the callout function, Destination Routing Descriptors in the DFSYDTx member of IMS.PROCLIB must be created specifying the appropriate properties for the callout destination. The properties depend on the type of callout.

We distinguish two types of callout requests:

- ▶ Asynchronous: The request is sent, but the sending IMS program does not wait for the response, which is returned as a separate IMS transaction invocation if necessary. This happens through an ISRT call to an ALTPCB destination.
- ▶ Synchronous: The sending IMS program waits for the response from the send / receive call flow. Also a time-out value needs to be provided. The request is performed through the new “ICAL” call function to an alternate destination, which is described through an OTMA descriptor.

A new optional keyword SYNTIMER is introduced to the OTMA descriptor so that the user can specify a time value to wait in hundredths of seconds, which can range from 1 to 900000, for synchronous call process to complete. See Example 11-1.

Example 11-1 OTMA descriptor keyword SYNTIMER

```
D WASDEST1 TYPE=IMSCON TMEMBER=HWS1 TPIPE=WASTP1 SYNTIMER=5000
```

11.4.1 Destinations for callouts through TM Resource Adapter

Two types of Enterprise Java Beans (EJB's) can be solicited for serving the callout request. They take the initiative to pull the callout message from an hold queue with a SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT interaction.

The types of EJB that we can use are:

- ▶ Stateless Session Bean (SLSB)

Stateless Session Beans are distributed objects, server-side Java EE components, that do not have state associated with them thus allowing concurrent access to the bean.

Stateless session EJBs display the following behavior:

- Provide a relatively short lived single use service.

- Do not maintain state on behalf of the client and do not survive EJB server crashes.
- Any two instances of the same stateless session EJB type are always identical, and each instance can be shared by multiple clients.

For more information about this subject refer to the WebSphere InfoCenter at URL:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/welcome_base.html

► **Message Driven Bean (MDB)**

Message-driven beans (MDBs) are stateless, server-side Java EE components for processing asynchronous messages. A MDB bean is an asynchronous bean activated by message deliveries and can consume and process messages concurrently.

A MDB can be used as a listener to receive callout requests that are retrieved by IMS TMRA. Then the MDB can route the message to the appropriate EJB or other application within the WebSphere Application Server for further processing.

11.4.2 Asynchronous callout requests

IMS application programs that are running in IMS-dependent regions can send outbound messages to request services from a WebSphere Application Server Enterprise Java Bean. The request for services or data is referred to as a callout request. When an IMS application program makes a callout request, IMS can be viewed as a client in a client-server relationship where the server is the external application to which IMS is making the callout request. If the request is treated as an asynchronous request, there will be no answer (send only), or a response is sent back as a new transaction.

An IMS configuration that supports callout includes these functions and components:

- The IMS application program that issues the callout request and If necessary, the IMS application program that processes the callout response.
- Optionally, an IMS database to store the data necessary to correlate the callout response to the initiating IMS client.
- An OTMA instance containing:
 - Asynchronous hold queue
 - OTMA ALTPCB destination descriptor
 - OTMA routing exits
 - OTMA asynchronous hold queue security
 - Optionally an OTMA Resume TPIPE Security exit routine (DFSYRTUX)
- IMS Connect instance
- WebSphere Application Server:
 - IMS TM Resource Adapter running on WebSphere Application Server
 - WebSphere Application Server EJB's
 - The external application program
- Optionally the use of RACF

Each Java application using IMS TMRA that processes a callout request externally can be configured to "listen" to the OTMA asynchronous hold queue by issuing a looping RESUME TPIPE call with an appropriate wait time. The Java application program might also be designed to preserve the correlation data and send it back with the callout response.

OTMA routing of callout requests is required to route the ISRT alternate_pcb call to the appropriate tpipe. The OTMA routing is performed by the OTMA ALTPCB destination

descriptor, or the OTMA Destination Resolution exit routine (DFSYPX0) and the OTMA User Data Formatting exit routine (DFSYDRU0).

In the WebSphere Application Server, the listening Java (EJB) code can be a Stateless Session Bean (SLSB) or the listener of a Message Driven Bean (MDB).

Securing the retrieval of asynchronous output and callout messages

The asynchronous output and the asynchronous callout programming models of the IMS TM Resource Adapter allow you to retrieve asynchronous output or callout request messages from the hold queue. To ensure that only an authorized user can retrieve an asynchronous output message or a callout request from the hold queue, you can optionally specify the user ID, together with the tpipe name that is contained in the SYNC_RECEIVE_ASYNCOUTPUT_* command message. Authorization is performed by IMS OTMA when the message is retrieved from the hold queue.

The user ID and password information can be specified in the IMSConnectionSpec object, in the authentication alias that the IMS ConnectionFactory uses, or the application's deployment descriptor. For more information about OTMA security, see the 'Specifying OTMA security' and 'Securing messages on the asynchronous hold queue' topics in *IMS Version 10 Communications and Connections Guide*, SC18-9703.

Asynchronous callout to a SLSB

Figure 11-6 shows the interaction with a Stateless Session EJB. Because the activity that is associated with the setup within the SLSB and IMS application are not directly tied together, we begin the flow not with the initiating Client, but within the SLSB running in WebSphere Application Server. This is opposite to the examples in Chapter 10, "The IMS SOAP Gateway" on page 203. If a callout request did not arrive to the waiting SLSB, it waits for the next available callout message or until a time-out occurs.

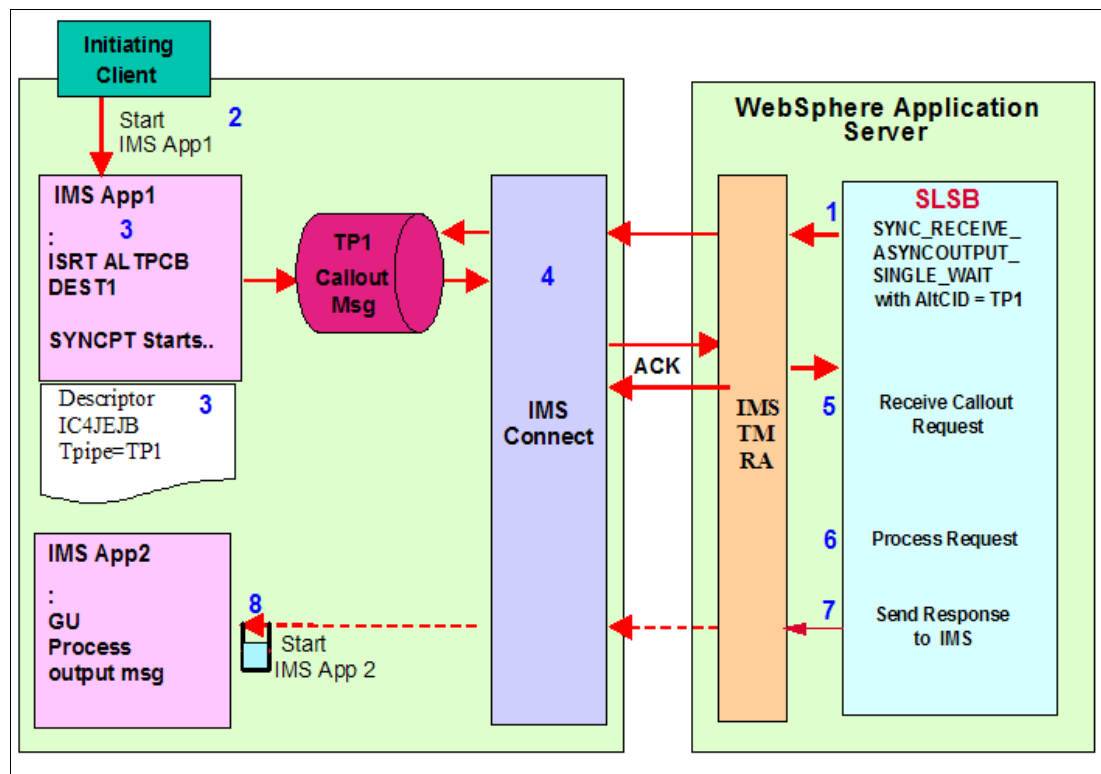


Figure 11-6 Message flow of Asynchronous Callout to SLSB using IMS TM Resource Adapter

The message flow of asynchronous callout to SLSB using the IMS TM Resource Adapter is:

1. In preparation for the flow, the EJB application in a WebSphere Application Server starts and obtains a shareable persistent connection to IMS Connect through the IMS TM Resource Adapter. It issues a SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT interaction, specifying the tpipe name as the value for the alternate client ID, and sets a large time-out value. TMRA in turn issues a RESUME TPIPE request to the tpipe and waits for the callout request from IMS Connect.
2. An initiating client, such as a terminal or an IMS Connect or OTMA client, causes an IMS application to start.
3. The IMS application issues an ISRT ALTPCB call to an OTMA destination routing descriptor, which contains the destination tpipe name. The callout request message is queued on this tpipe.
4. As soon as the callout request is available in the tpipe, IMS Connect delivers the callout message to TMRA.
5. TMRA receives the callout request message and presents the callout request to the EJB.
6. The EJB processes the callout request.
7. If the EJB has response data to be sent back to IMS, the EJB issues a normal IMS send_only transaction request through IMS TMRA and IMS Connect to IMS.
8. A second IMS application is scheduled to handle the asynchronous reply.

To prepare a Java application for inbound requests from an IMS application program, you must specify the appropriate interaction verb and the asynchronous hold queue name in the Java application. Also, specify a time-out value.

Specify the asynchronous hold queue name.

```
InteractionSpec.setAltClientID(calloutQueueName);
```

Set the interactionVerb property to SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT.

```
interactionSpec.setInteractionVerb(com.ibm.connector2.ims.ico.  
IMSInteractionSpec.SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT);
```

Specify an execution time-out value (in milliseconds) that you want to wait for a callout request message in the hold queue.

```
interactionSpec.setExecutionTimeout(3600000);
```

In this example, we specify a large execution time-out value. A large execution time-out value helps to minimize the looping required in a callout EJB when there might be long periods of time when no callout requests occur.

Sample code related to the asynchronous callout to a SLSB can be found in “Asynchronous callout to a Stateless Session Bean” on page 355.

Asynchronous callout to an MDB

Message Driven Beans (MDB) allow your application to asynchronously receive messages delivered to a JMS destination. Figure 11-7 on page 238 presents the interaction with a Message Driven Bean.

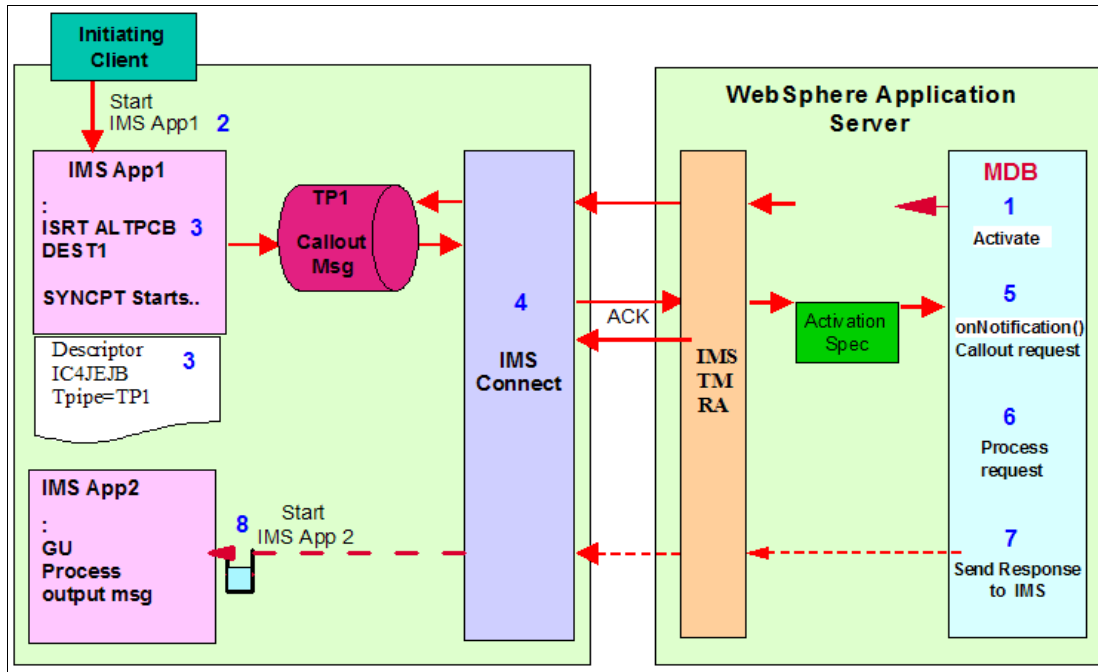


Figure 11-7 Message flow of asynchronous callout to MDB using TMRA

The message flow of an asynchronous callout to MDB using TMRA is:

1. When the Java EE application containing the deployed MDB is started in the WebSphere Application Server through the *ActivationSpec* and the underlying listening software, a shareable persistent connection to IMS Connect through TMRA is obtained. The Java EE application issues a `SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT` interaction, which specifies the tppe name as the value for the alternate client ID and sets a large time-out value. TMRA, in turn, issues a `RESUME TPIPE` request to the tppe and waits for the callout request from IMS Connect. The addressing information is pulled from properties in the EJB(MDB) deployment descriptor.
2. An initiating client, such as a terminal, an IMS Connect, or OTMA client, provokes the scheduling of an IMS application.
3. The IMS application issues an `ISRT ALTPCB` call to an OTMA destination routing descriptor, which contains the destination tppe name. The callout request message is queued in this tppe.
4. As soon as the callout request is available in the tppe, IMS Connect delivers the callout message to TMRA.
5. TMRA receives the callout request message and passes the request to the MDB in the *onNotification()* method.
6. The MDB processes the callout request.
7. If response data must be sent back to IMS, the MDB issues a normal IMS transaction with a `SYNC_SEND` request to the appropriate IMS application with the trancode and response data.
8. In this case, IMS App2 is initiated by the receipt of the response from the MDB.

Sample MDB code is in “Asynchronous callout to a Message Driven Bean” on page 357.

11.4.3 Synchronous callout requests

Figure 11-8 shows that with the IMS callout support, IMS applications can be a client and server. IMS provides bi-directional access between IMS applications and external application and servers. The IMS application program can:

- ▶ Callout to user-written IMS Connect client
- ▶ Callout to WebSphere EJB/MDB using IMS TM Resource Adapter
- ▶ Callout to Web Service Provider using IMS SOAP Gateway

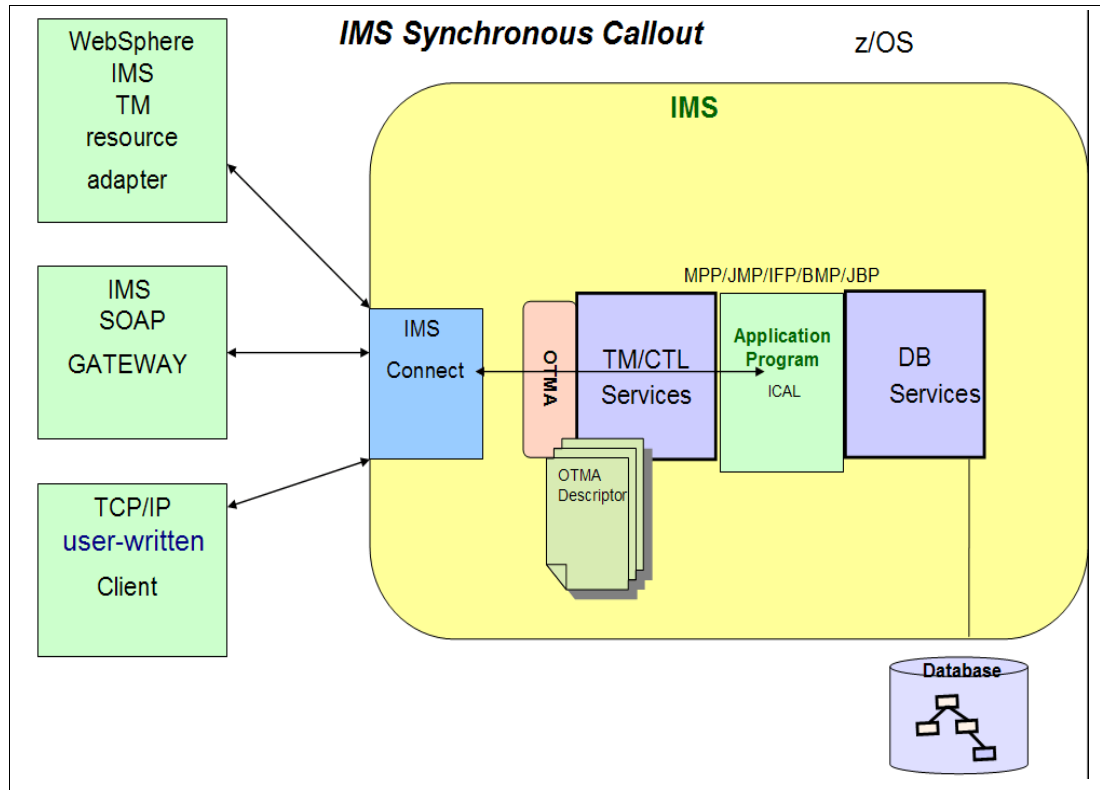


Figure 11-8 Synchronous Callout interfaces

A new DLI call, the ICAL SENDRECV synchronous call function, allows an IMS application program to communicate with other external applications synchronously. An IMS Call (*ICAL*) sends a synchronous request for data or services from an IMS application program running in the IMS TM environment to a non-IMS application program or service running in a distributed environment. ICAL processing does not use the IMS message queue.

Here is the format of this new DLI call:

```
CALL 'AIBTDLI' USING function, aib,i/o area
```

The parameters are:

- ▶ function: set to "ICAL"
- ▶ aib

Specifies the application interface block (AIB) to be used for this call.

- AIBID
Eyecatcher. This 8-byte field must contain DFSAIBbb.
- AIBLEN

AIB length. This field must contain the actual length of the AIB that the application program obtained.

– AIBSFUNC

This sub function code field must contain one of the listed 8 byte sub function codes: SENDRECV. The IMS application program uses this sub function for synchronous program to program communication.

– AIBRSNM1

This resource name field contains the OTMA descriptor name. AIBRSNM1 is an 8 byte alphanumeric left-justified field padded with blanks.

– AIBRSFLD

This 4 byte field is used by the IMS application program to specify a time value to wait in 100th of a second for the synchronous call process to complete. This value overrides the value specified in the OTMA descriptor. The minimum value is 0 and the maximum value is 999999. The system default is 10 seconds.

– AIBOALEN

This request area length is an input and an output parameter. As an input parameter this 4-byte field must contain the length of the input request area that is specified in the call list. As an output parameter this field is only updated when partial data is returned (AIB return code x'100', AIB reason code x'00C'). In the case of partial data returned, this field contains the actual length of the response message. For any return code other than partial data, this field is unchanged.

► i/o area

The ICAL synchronous call function provides the ability for an IMS application program to communicate with other application programs in a synchronous manner using the new OTMA message processing function. Because the IMS message queue is not used, the 32K message segment restriction was removed.

The ICAL synchronous call function is only supported using the AIBTDLI interface in IMS TM runtime environments for IFP, MPP, BMP, JMP, and JBP regions.

Attention: Coordinated two phase commit between the IMS application program and the external application program is not supported in the IMS Version 10 SPE. Also a Shared Queues back-end IMS which is not connected to IMS Connect, cannot issue this ICAL.

The functions that OTMA provides as part of its support for the IMS synchronous callout Version 10 SPE are:

- Flow the synchronous callout request and receive the response
- Enhance the OTMA output descriptor
- Provide the time-out function for the ICAL call
- Provide the OTMA protocol updates so that the synchronous callout request and the response can be recognized by both IMS Connect and OTMA
- Provide a simplified OTMA TPIPE log record to track input and output flow of a synchronous callout message
- Provide the OTMA command updates

OTMA connects the ICAL call to the IMS Connect client applications so that a synchronous callout request can be delivered to them and the response can be given back to the IMS application. This synchronous callout request is delivered through the OTMA Resume TPIPE

method. If the request is larger than 32K, it is delivered to IMS Connect using a multi-segment flow. The size of each message segment is equal or less than 32K. IMS Connect assembles the segments and deliver them as one complete message to IMS Connect client applications, such as IMS TM Resource Adapter.

Synchronous callout to an SLSB

Figure 11-9 shows the interaction with a Stateless Session EJB.

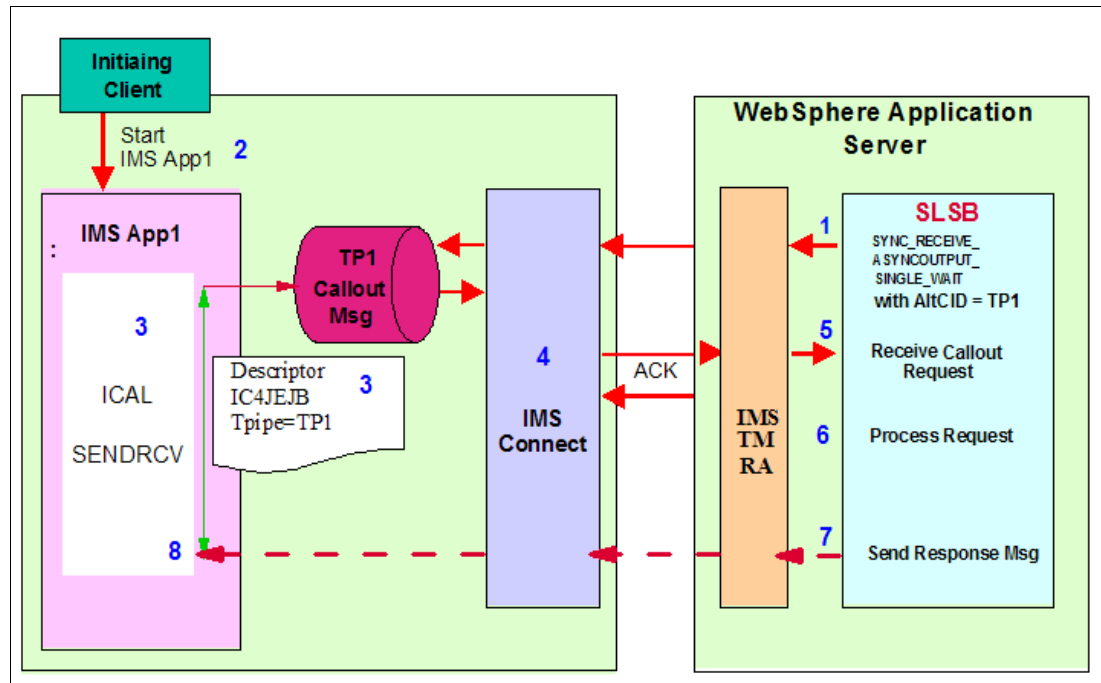


Figure 11-9 Message flow of synchronous callout to SLSB using TMRA

The message flow of a synchronous callout to SLSB using TMRA is:

1. The SLB application in WebSphere Application Server starts and obtains a sharable persistent connection to IMS Connect through TMRA. It issues a SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT interaction, specifying the tpipe name as the value for the alternate client ID, and sets a large time-out value. TMRA in turn issues a RESUME TPIPE request to the tpipe and waits for the callout request from IMS Connect.
2. An initiating client, such as a terminal, an IMS Connect, or an OTMA client, starts an IMS application.
3. The IMS application issues an ICAL call to an OTMA destination routing descriptor, which contains the destination tpipe name. The callout request message is queued in this tpipe.
4. As soon as the callout request is available in the Tpipe, IMS Connect delivers the callout message to the IMS TM Resource Adapter.
5. TMRA receives the callout request message and returns the callout request to the SLSB.
6. The EJB processes the callout request.
7. A response message for a synchronous callout request return to OTMA as a special form of SYNC_SEND message. This type of Send-Only message has no trancode and can be a regular response data or error information to be passed back to the ICAL call. The response message can also be a multi-segment message. OTMA assembles the

multi-segment messages into a single message to be returned back to the IMS application.

The correlation between request from IMS and response to IMS, is assumed by the correlation token in *interactionSpec* of the SYNC_RECEIVE_ASYNCOUPTUT_SINGLE_WAIT. and SYNC_SEND interaction.

8. The waiting IMS App1 is presented the response.

Sample code for the SLSB is located in “Synchronous callout to a StateLess Session Bean” on page 360.

Synchronous callout to an MDB

Message Driven Beans allow your application to synchronously receive messages delivered to a JMS destination. Figure 11-10 presents the interaction with a MDB.

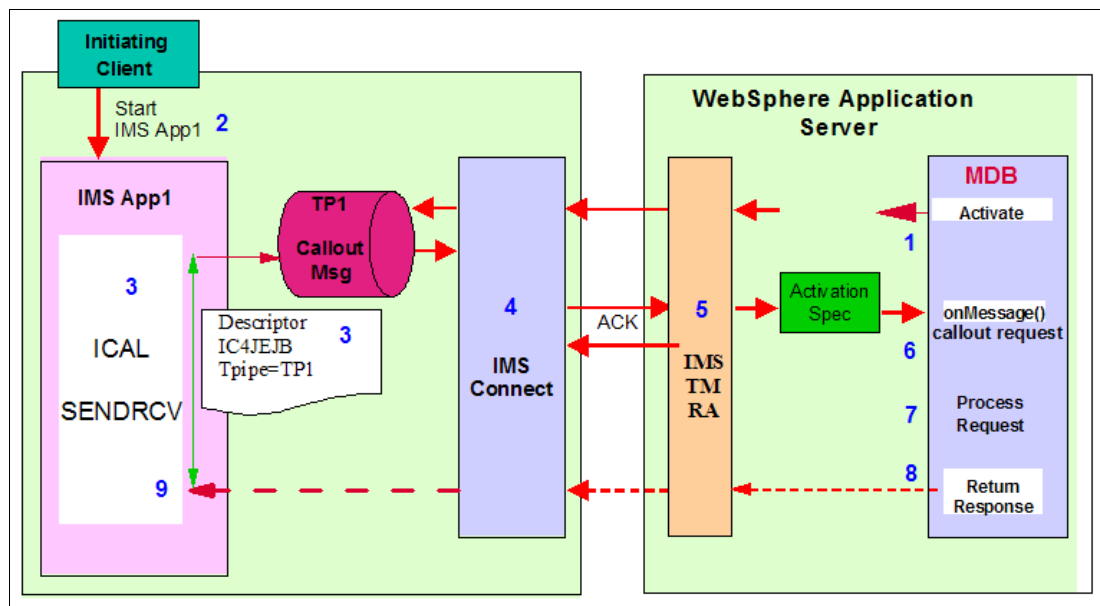


Figure 11-10 Message flow of Synchronous Callout to MDB using IMS TM Resource Adapter

The message flow of a synchronous callout to MDB using IMS TM Resource Adapter is:

1. When the Java EE application that contains the deployed MDB is started in the WebSphere Application Server through the *ActivationSpec* and the underlying listening software, a sharable persistent connection to IMS Connect is obtained through TMRA. It issues a SYNC_RECEIVE_ASYNCOUPTUT_SINGLE_WAIT interaction, specifying the tpipe name as the value for the alternate client ID, and sets a large time-out value. TMRA issues a RESUME TPIPE request to the tpipe and waits for the callout request from IMS Connect. The addressing information is pulled from properties in the MDB deployment descriptor.
2. An initiating client, such as a terminal, an IMS Connect, or an OTMA client provokes the scheduling of an IMS application.
3. The IMS application issues an ICAL call to an OTMA destination routing descriptor, which contains the destination tpipe name.
4. The callout request message is queued in this tpipe.
5. As soon as the callout request is available in the tpipe, IMS Connect delivers the callout message to TMRA.

6. TMRA receives the callout request message and passes the callout request to the MDB in the `onMessage()` method.
7. The MDB processes the callout request.
8. Response data are returned as an object of type *Javax.resource.cci.Record* with the output data.
9. The waiting IMS App1 is presented the response.

Sample code for the MDB is located in A.4, “Synchronous callout to a Message Driven Bean” on page 361.

This concludes our discussion about the IMS Transaction Manager Resource Adapter.



MFS Web solutions

IMS MFS Web Solutions consists of IMS MFS Web Enablement and IMS MFS SOA support. MFS Web Solutions is designed so that MFS-based IMS applications can be retargeted to support B2B and B2C XML communications. Both of these solutions provide customer access to existing MFS-based IMS business logic in the Web world, achieving flexibility and reusability.

In this chapter, we discuss:

- ▶ Technologies behind MFS Web Solutions
- ▶ IMS MFS Web Enablement
- ▶ IMS MFS SOA support

12.1 Technologies behind MFS Web Solutions

There are a few important technologies that are fundamental to MFS Web Solutions, and they are:

- ▶ XML: Extensible Markup Language is growing in acceptance as the universal data format for any application. XML, as a logical choice, is chosen to represent the information in MFS control blocks to the world of MFS Web Solutions.

When an XML request is received, it is transformed to a byte stream by retrieving the relevant information from the MFS XML repository. The input byte stream can then be placed in an IMS message queue to await processing by an MFS-based IMS application program. A byte stream response is generated by the MFS-based IMS application and is transformed into an XML response, again by retrieving the relevant information from the MFS XML repository.

- ▶ MFS Common Application Metamodel: MFS Common Application Metamodel (CAM) is used by the MFS Importer to capture information that exists in the MFS source files, which includes:
 - Input and output message fields
 - Display information
 - MFS flow control
 - Device characteristics
 - Operation semantics

The MFS importer parses the MFS source to produce MFS Web artifacts.

Figure 12-1 on page 247 shows the metamodel mapping of the MFS source relationships.

MFS Web Services Business-to-Business:

- ▶ No VTAM or 3270 emulator required
- ▶ Integrated with WebSphere tooling

MFS Web Enablement Business-to-Consumer:

- ▶ No tooling, VTAM, or 3270 emulators required
- ▶ Uses modern Web browser functionality

12.1.2 Sample Client experience with IMS Web Solutions

Figure 12-2 illustrates a telephone control system using a Windows-based application as a front end to interface with IMS. In this system, incoming customer calls are intercepted by a phone switch and passed to the Windows-based application, which triggers a pop-up window that contains a pre-populated customer profile, including sales, invoice, and call history data. This information appears before each call is answered by a sales person, providing them with vital customer service information.

The customer data is retrieved from Oracle® tables that were refreshed nightly through downloads. With this design, Figure 12-2, extra work is required to maintain multiple disparate systems, and there is no real time access to current critical data. Essentially this is an inefficient and non integrated design.

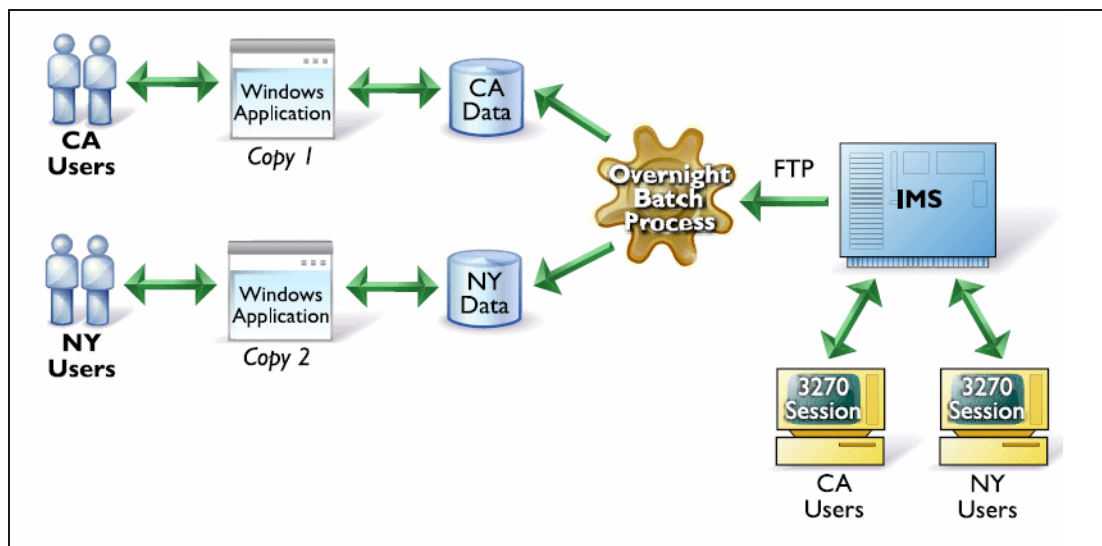


Figure 12-2 Traditional Windows-based application front-end interface to IMS

To improve the speed and efficiency of the overall process, a more flexible architecture is needed for real time access to the Oracle tables. A transition from a Windows based platform to a Web services-based application environment can be used to meet this goal. This transition does not require a complete system redesign because there are no changes to the back end IMS transaction and application set. There are requirements to modernize the customer-facing front end, and integrate it with the existing back end environment.

The solution uses a Java-based Web application, and connects with the customer interaction infrastructure. The system accesses the centralized application and database servers and communicates with the back-end IMS system. Existing IMS transactions are accessed from the Web application without any required programming changes by running IMS TM Resource Adapter on WebSphere Application Server, with IMS Connect running at the host.

IMS transaction-level security is maintained using RACF-managed userIDs and passwords (specified as IMSConnectionSpec properties).

Figure 12-3 shows the integrated design with MFS SOA Support.

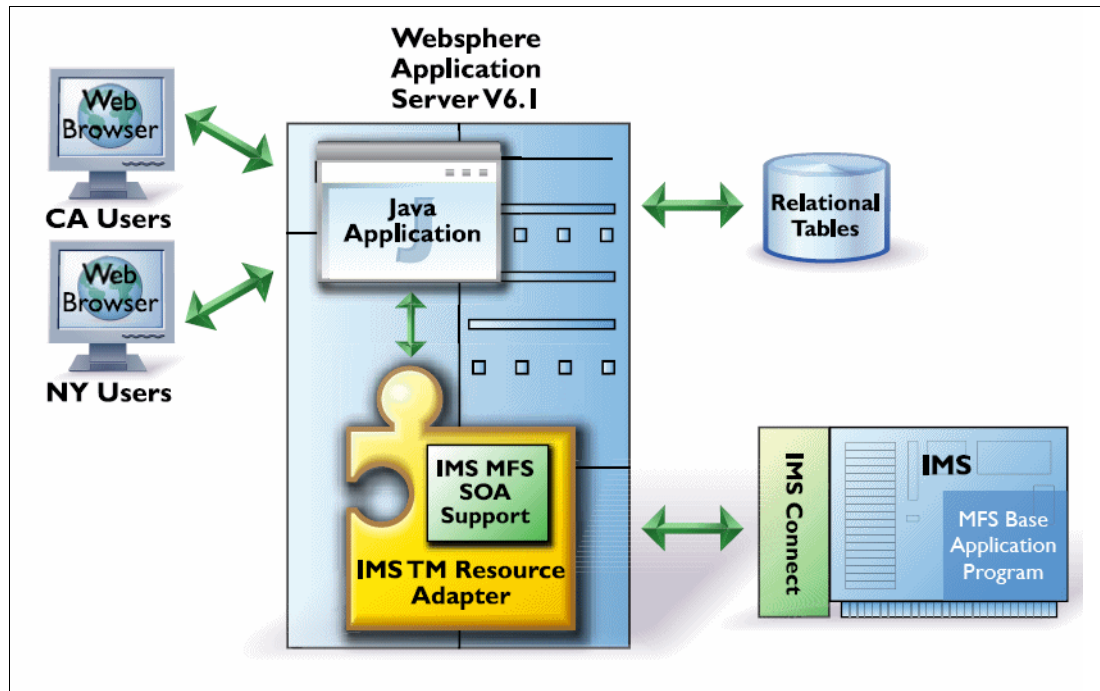


Figure 12-3 MFS SOA Support integration

By using IMS MFS Web Services, existing business logic in the evolving Business-to-Business environment can be accessed without using VTAM and 3270 emulators, which helps to reduce the total cost of ownership while also revitalizing existing 3270 based transactions.

12.2 MFS Web Enablement

Using IMS MFS Web Enablement you can reuse existing MFS-based IMS business logic from the Web. The MFS Importer, available in the stand alone MFS XML utility, generates metadata from existing MFS source files during the development phase. The runtime support operating on a WebSphere Application Server provides a servlet to dynamically render MFS-based Web pages on browsers using cascading stylesheets, and an adapter that communicates with host IMS (conversational) applications through the Java EE compliant IMS TM Resource Adapter. Sample stylesheets are provided and can be customized to fit your business needs.

In summary, IMS MFS Web Enablement Version 9.3 provides the tooling utility and runtime support to Web-enable existing or new IMS MFS-based applications in IBM WebSphere Application Server, and interactively render them for display in standard browsers, such as Microsoft Internet Explorer and Mozilla Firefox.

Figure 12-4 on page 250 presents an overview of the MFS Web Enablement architecture.

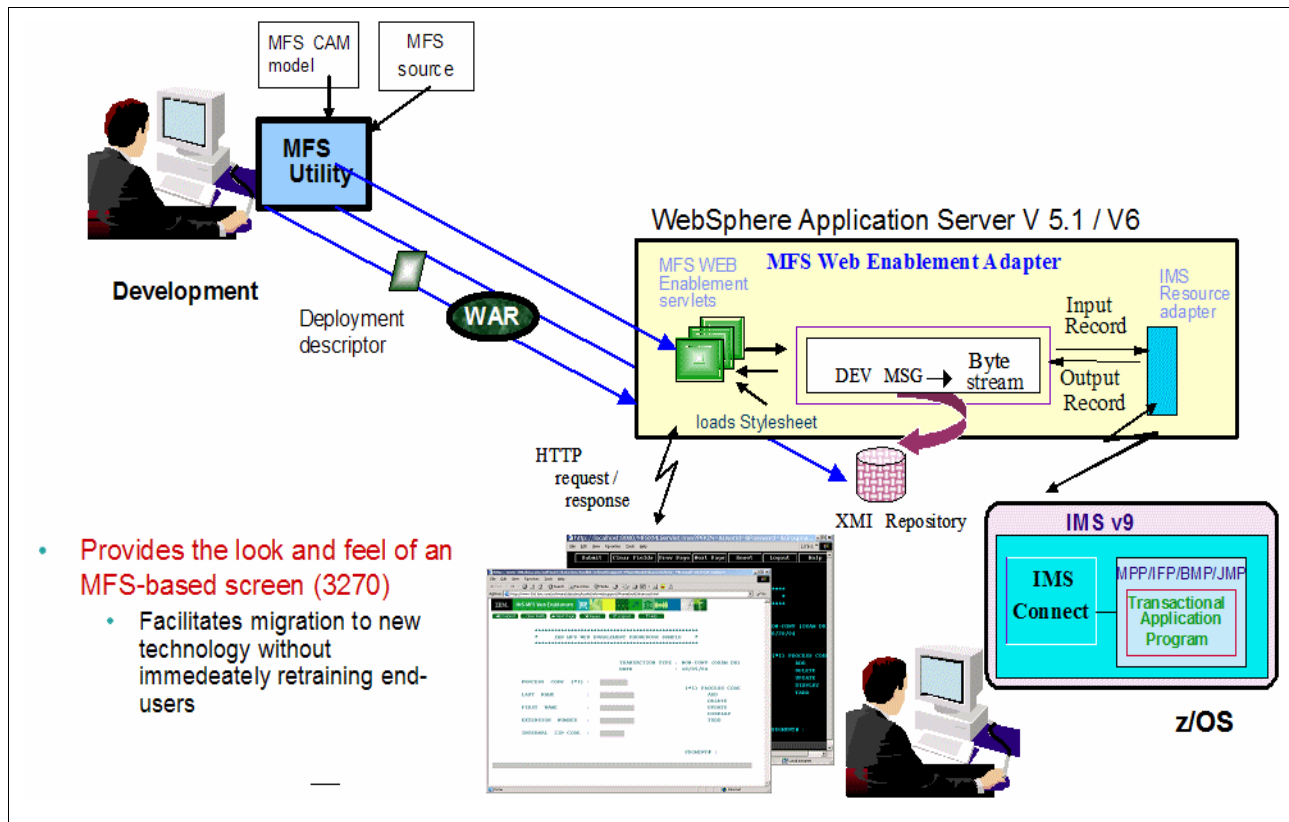


Figure 12-4 MFS Web enablement architecture

12.2.1 Prerequisites for MFS Web Enablement

The platforms that support IMS MFS Web Enablement Version 9.3 include WebSphere Application Server for Microsoft Windows, AIX, and z/OS.

You must have the following products and tools installed to use MFS Web Enablement Version 9.3.0:

- ▶ IMS Version 9.1 or beyond
- ▶ IMS Connect Version 9.1.0.x or beyond
- ▶ IBM WebSphere Application Server distributed platforms (Windows and AIX only) Version 5.1 or 6.0, or IBM WebSphere Application Server z/OS Version 6.0 with the latest fix packs applied. Version 5.1.1 of WebSphere Application Server requires both the cumulative fix pack and the cumulative fix pack for SDKs. WebSphere Application Server Version 6.1 for Windows distributed is also supported.
- ▶ IMS TM Resource Adapter 9.1.0.x
- ▶ One of the following Web browsers:
 - Microsoft Internet Explorer Version 6 or later
 - Mozilla Firefox Version 1.0.1 or later

12.2.2 How does MFS Web Enablement work

IMS MFS Web Enablement support provides Business-to-Consumer solutions to reuse customer's existing MFS-based IMS applications. It is designed for users who are want close

adaptation of the original 3270 MFS-based IMS transactions on the Web but do not want to rely on 3270 data streams-based solutions. It is also designed for users who do not wish to rely on WebSphere based development tooling such as Rational Application Developer or WebSphere Integration Developer.

Consider MFS Web Enablement if there exists a possible requirement to execute MFS based IMS transactions on small devices, such as PDA or Apple's iPhone.

During the development phase, you use the IMS MFS XML utility to invoke the MFS Importer to parse MFS source files of existing MFS-based IMS transactions and generate Java classes and XMI metadata files for runtime processing. The Java classes are packaged into a Web application archive (WAR) file and deployed on WebSphere Application Server. XMI Metadata files are used during runtime for data transformation.

During the runtime phase, the HTTP request from a browser is sent to the Web application that runs on the application server, which performs in conjunction with the IMS MFS Web Enablement runtime component. If this request is the initial request, a new HTTP session is created and the initial Web page is returned, simulating the 3270 type terminal blank panel. Subsequent requests that contain input data (which can be transaction or command codes), are transformed into an input byte stream to be sent across to the IMS host application. The output byte stream that comes back from the application is transformed into XML data objects that are stored in the HTTP session. The XML data object contains one or more physical pages to be displayed as Web pages. Web pages are returned in the HTTP response, one page per 3270-PA1 equivalent paging request.

The MFS servlet receives an HTTP request and loads the session objects into memory. If this request is the initial request, the servlet creates a new session and sends out the initial blank page, which is a representation of the 3270 type terminal blank panel, for display, for example:

<http://servername:portnumber/contextroot/servletname>

To assist in understanding these processes, Figure 12-5 on page 252 presents the MFS Web Enablement runtime environment.

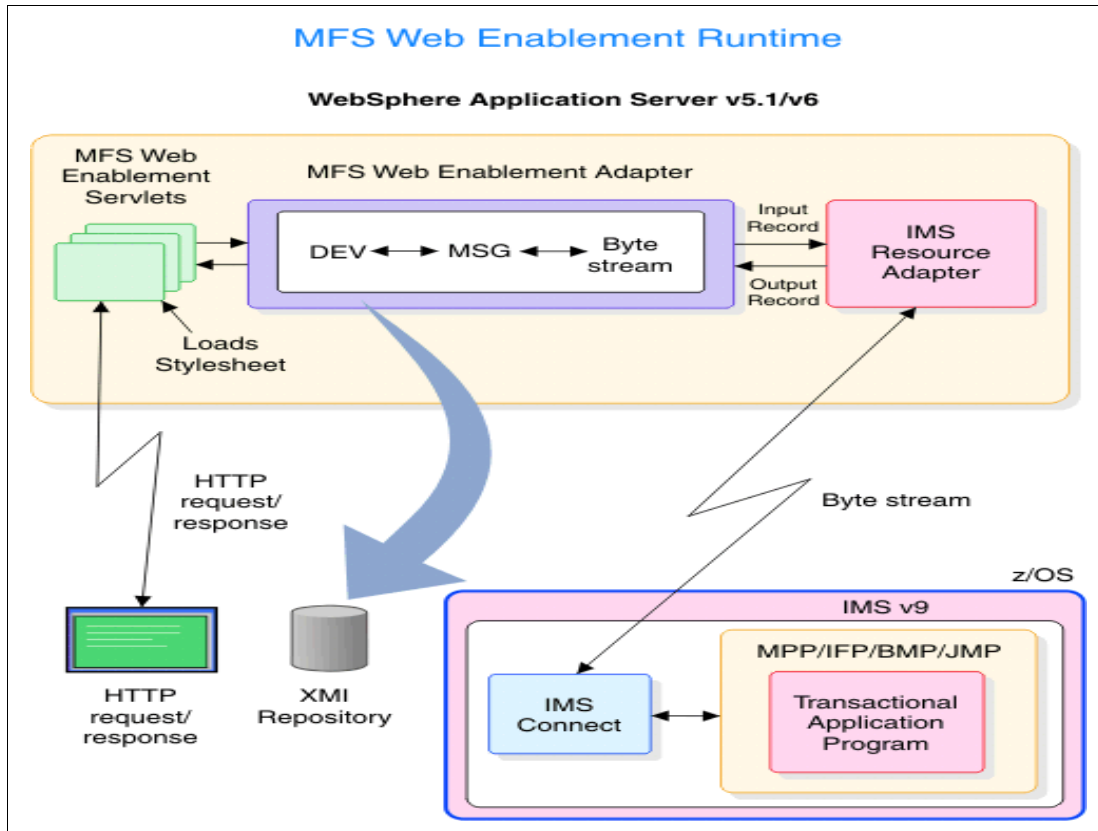


Figure 12-5 The MFS Web Enablement runtime environment

12.2.3 The MFS Adapter component of MFS Web Enablement

The MFS Adapter runs inside WebSphere Application Server and transforms data between MFS device and message data. The MFS Adapter is invoked by the MFS Servlet. Using the Eclipse modeling framework, the MFS Adapter loads the appropriate MFS XMI resource from the repository, invokes the transformer routine to handle the data conversion, and submits the IMS transaction using the TM RA Common Client Interface (CCI) method calls.

Based on the information that is contained in the DIF/MID XMI file, the transformer routine first maps the input device data into message data and then into an input byte array. The input byte array is sent across using TMRA. Upon successful execution, the output byte array comes back on the return route. The MFS Adapter then loads the DOF/MOD XMI file, specified in mapName and invokes the transformer routine to first map the output byte array into message data, and then into output device data. The resulting data object is returned to the MFS Servlet.

Figure 12-6 on page 253 through Figure 12-12 on page 255 show how the MFS Adapter handles the data transformation.

Figure 12-6 on page 253 displays that when the format request initially is presented, a lookup from the XMI repository for the specified MOD XMI is performed by the MFS Adapter.

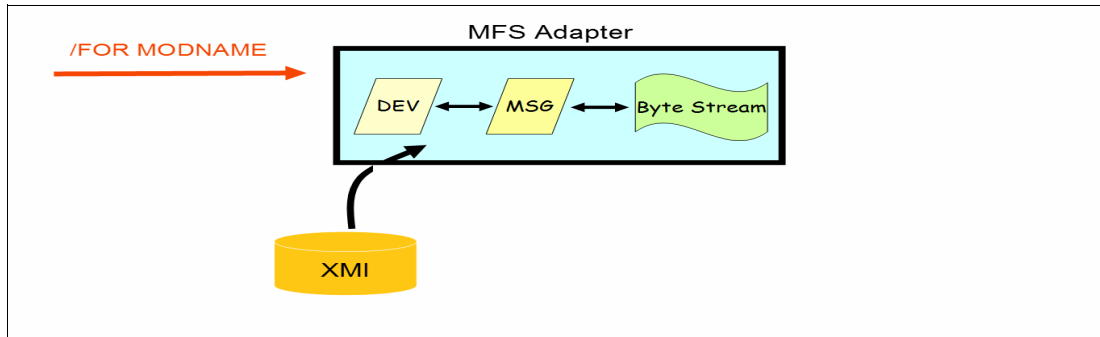


Figure 12-6 XMI lookup for the MOD XMI

Figure 12-7 shows the response of a formatted page returning to the client.

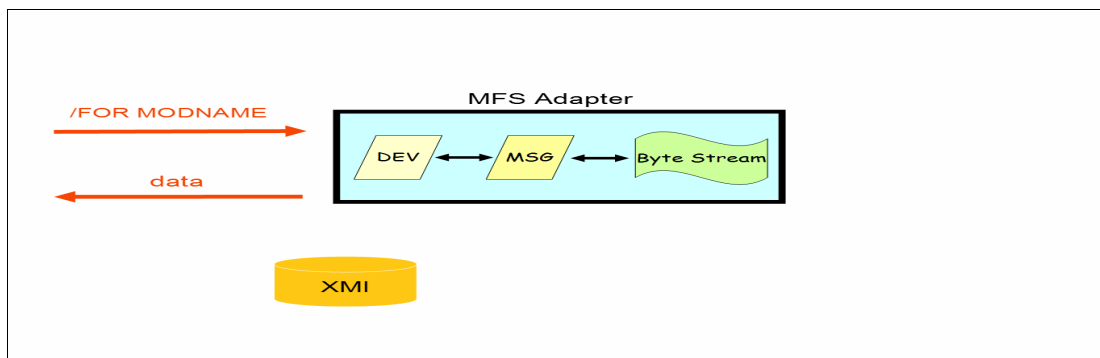


Figure 12-7 Response to the initial format request

Figure 12-8 displays that when an input transaction and data is presented, a lookup from the XMI repository is performed to handle the device to message side mapping and transforms the data to byte stream format.

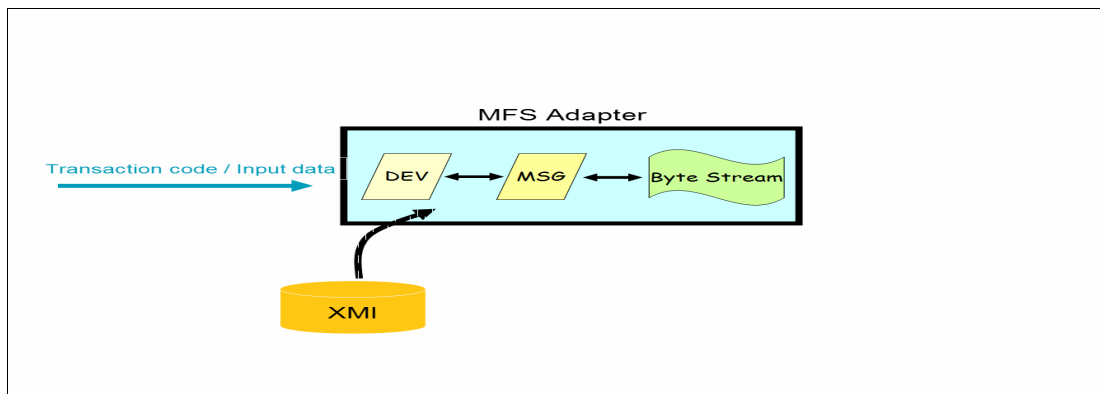


Figure 12-8 Lookup for first input transaction code and data flow

Figure 12-9 on page 254 shows the input transaction code and data and the next message MOD name in the control flow that is being passed to the application.

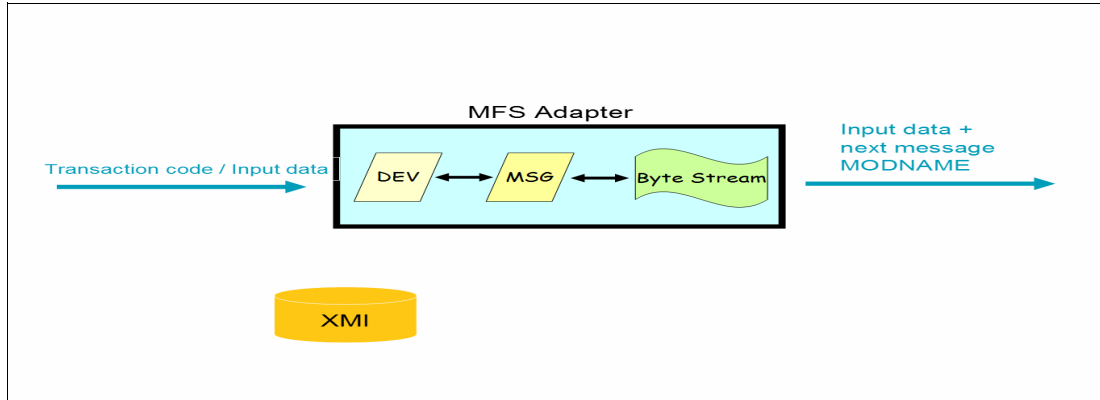


Figure 12-9 Transferring input data and MOD name in preparation of the next output message

Figure 12-10 displays that when output data is returned, a lookup in the XMI repository occurs to handle the message to device side transformation.

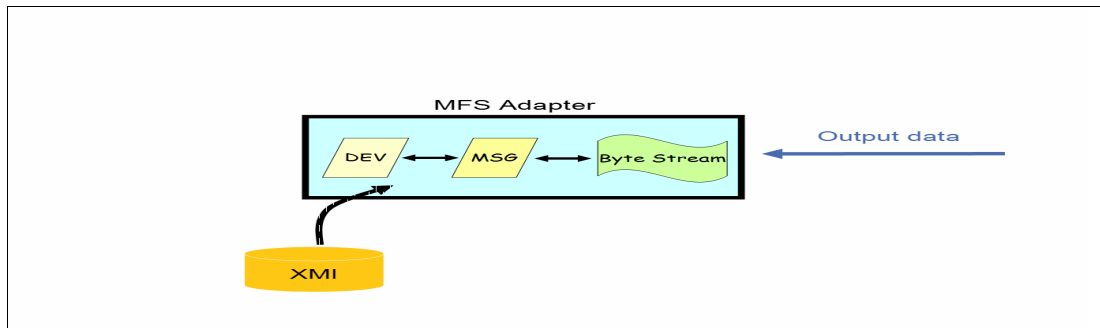


Figure 12-10 Message to device side transformation

Figure 12-11 illustrates the data being returned to the client.

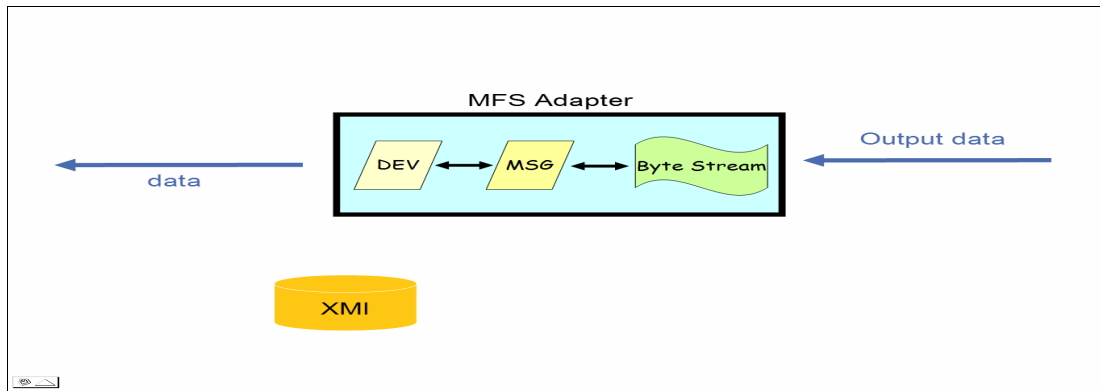


Figure 12-11 Output data flow through the MFS Adapter

Figure 12-12 on page 255 displays the scenario when the application changes the MOD name.

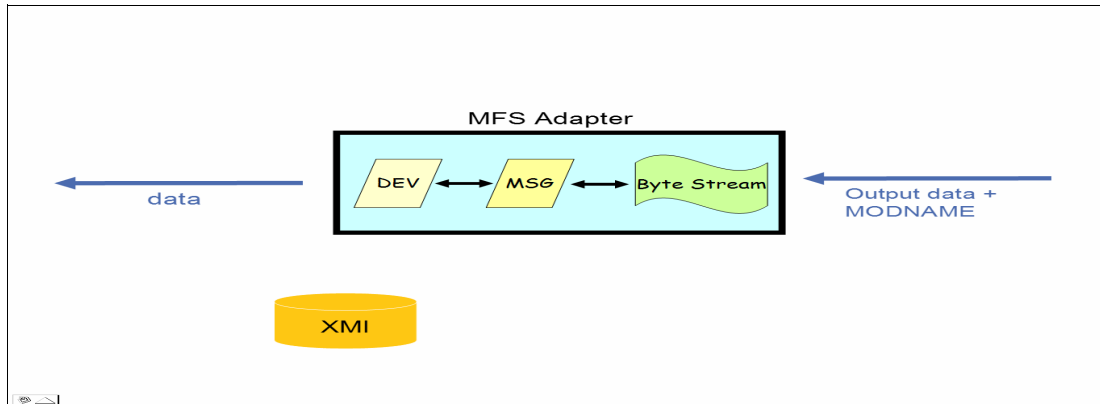


Figure 12-12 Situation when the application changes the MOD name

12.2.4 MFS Servlet and stylesheets

The MFS Servlet is the super class of all MFS instance servlets. The MFS Servlet runs in the WebSphere Application Server and handles HTTP requests and responses to and from client browsers. The MFS Servlet is responsible for connection state management, interaction with the MFS Adapter, and rendering of MFS XMI objects using the stylesheet to dynamically produce MFS Web pages.

Figure 12-13 shows the interaction of the MFS Servlet along with instance servlets and the client.

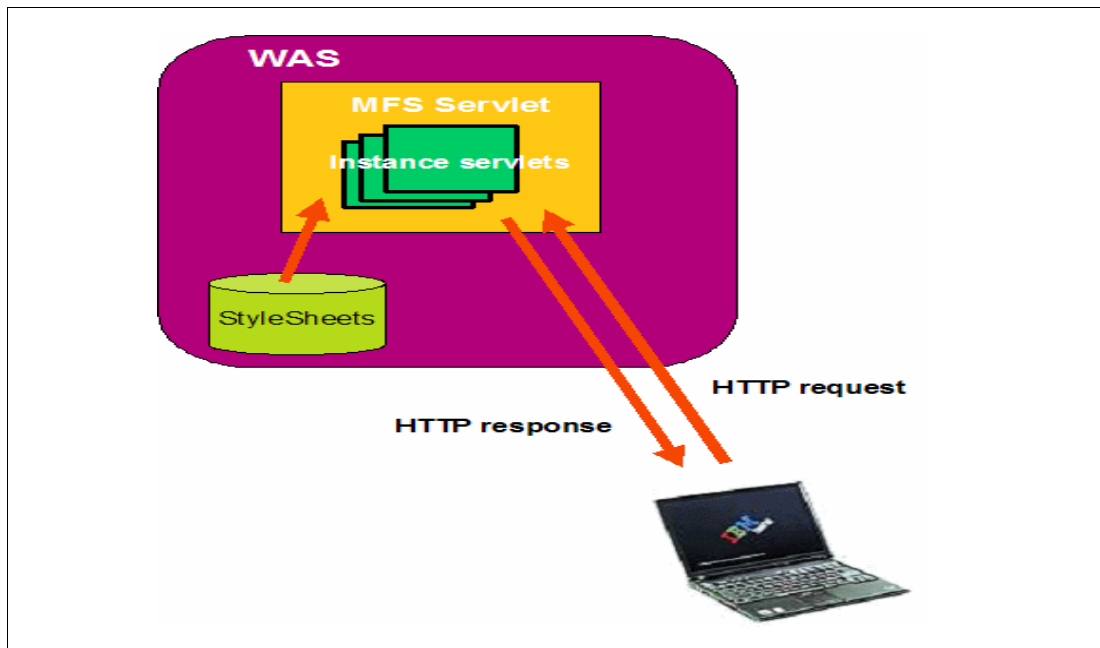


Figure 12-13 MFS Servlet and instance servlets and clients

The instance servlet, which is generated from the MFS XML utility, extends the MFS Servlet. The instance servlet name and initialization parameters are recorded in the web.xml file in the WAR file.

Upon receiving the HTTP request, the MFS Servlet retrieves the state information for the client.

12.2.5 The MFS XML utility

The MFS XML utility is a command line development tool that runs on a Microsoft DOS command prompt. This utility generates all of the necessary files that are needed to Web enable your MFS-based IMS applications. It takes MFS source files as input and produces metadata XMI files and WAR files as output. In addition, the utility provides FTP client support to transport the generated output to the WebSphere Application Server.

Step1: Invoking MFS XML utility

There are several steps that are associated with the use of the MFS XML utility. But first we discuss the role the IMS MFS Importer performs.

The IMS MFS Importer reads and parses MFS source files for an application and generates XMI instance files that describe the MFS-based application interface. The XMI file represents all of the application interface information that is encapsulated by the MFS source, which includes the input and output device descriptors, message descriptors, MID MOD relationship, device characteristics, and operation semantics. To ensure non-proprietary access, the MFS Importer is built using the MFS Common Application Metamodel, as mentioned in 12.1, “Technologies behind MFS Web Solutions” on page 246.

You invoke the MFS XML utility by starting the mfsxml.bat file using an MS-DOS-based command prompt window.

Step 2: Generating XMI files from MFS source files

The MFS XML utility invokes the MFS Importer and uses the Eclipse Modeling Framework to serialize each MID/DIF pair, MOD/DOF pair, and MFS TABLE into an XMI file. The generated XMI files are transferred to an XMI repository on WebSphere Application Server and are read for data transformation during runtime.

The optional device characteristics table file specifies the window size of certain device types. Transfer the file in binary format from MVS to the system running the MFS XML Utility. You can transfer the MFS source files from MVS in either text or binary format. Because of this, you must indicate which mode you are using to transfer the files.

The MFS Importer is invoked after you specify the host codepage. If the parsing of a specific MFS source file results in a warning, the XMI files are still generated. However, if the parsing resulted in errors, the XMI files are not generated, and you return to the MFS XML utility menu.

After the XMI files are parsed, you are prompted to select a device type and feature. The output directory specifies where to save the generated XMI files on the local system. Note that you can run through step 2 multiple times to generate different connection settings.

Step 3: Generating the instance servlet and the web.xml files

You generate the instance servlet class files and web.xml files for packaging into a Web application archive file.

Step 4: Generating the Web application archive file

The MFS XML utility generates a Web application archive file from one or more instance servlets and the web.xml file that you generated in step 2.

The MFS XML utility packages the generated instance servlets and deployment descriptor web.xml files into a Java EE compliant WAR file that is deployable on WebSphere Application Server. The deployment web.xml file stores a specific stylesheet, XMI repository, host

connection information, and time-out settings that are related to TMRA. The Java EE-compliant WAR file contains one or more Java files, class files, and web.xml files.

Step 5: Uploading WAR and XMI files using an FTP client

In the case where your MFS XML utility is running on a different machine than your WebSphere Application Server, the MFS XML utility provides a FTP client to upload the WAR and XMI files onto the WebSphere Application Server. To use the FTP client, you must know the host name, user ID, password, server-side directory path, and the directory that contains the files to upload.

Step 6: Running a batch file (optional)

The MFS XML Utility lets you use a previously-saved batch file in “Step 2: Generating XMI files from MFS source files” on page 256 and “Step 3: Generating the instance servlet and the web.xml files” on page 256, instead of repeatedly invoking these steps.

Figure 12-14 shows the summary of artifacts that the MFS XML utility generates. Although it shows both the Windows workstation and the server environment, the MFS XML utility and WebSphere Application Server can be on the same machine.

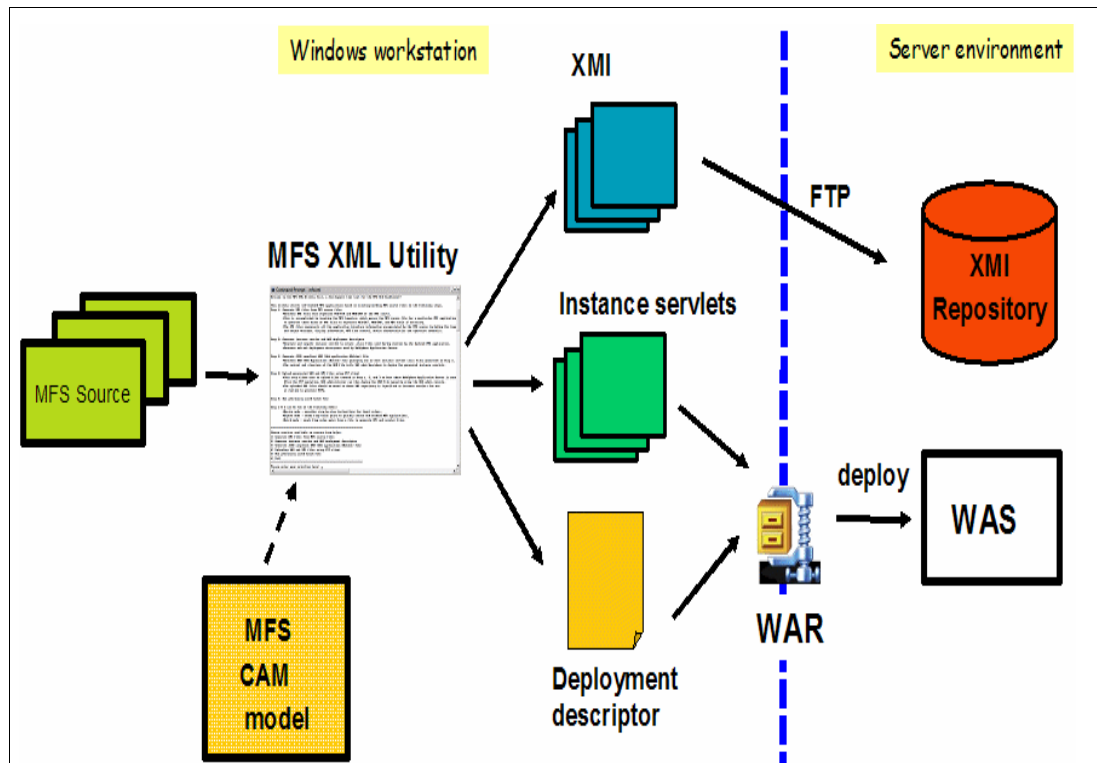


Figure 12-14 Artifacts that the MFS XML utility generates

12.2.6 Running MFS Web Enablement

Upon receiving the HTTP request, the MFS Servlet checks if this request is associated with an existing HTTP session. If a match is not found, the HTTP session is created and the initial Web page, simulating the 3270 blank panel, is generated and returned.

From the initial Web page, you can enter the inputs in the next section.

RACF user ID, password, and group name

The RACF credentials that the user specified while running the MFS XML utility are displayed as the default value. You can choose to supply a different set of credentials. The supplied credentials are valid for the entire session. The session is terminated after you log out, close the browser, or when the session times out. When new credentials are specified, the previously active connection and pending conversation are automatically terminated, and a new TMRA connection object is created. The MFS Adapter converts the RACF user ID, password, and group name to uppercase text.

The /FOR or /FORMAT modname command

The MFS Adapter processes the format command, which attempts to load the MOD /DOF XMI file, based on the modname, from the XMI repository. The MFS Servlet then renders the DOF metadata with the MFS stylesheet and returns the formatted Web page to the client browser. If the specified modname is not found, the system returns the message:

```
IXFT003E: REQUESTED XMI NOT FOUND: MODNAME using system default DFSM03.xmi.
```

Figure 12-15 displays the output of the formatted IMS IVP Phonebook panel that is obtained by issuing “/FOR IVTNO”.

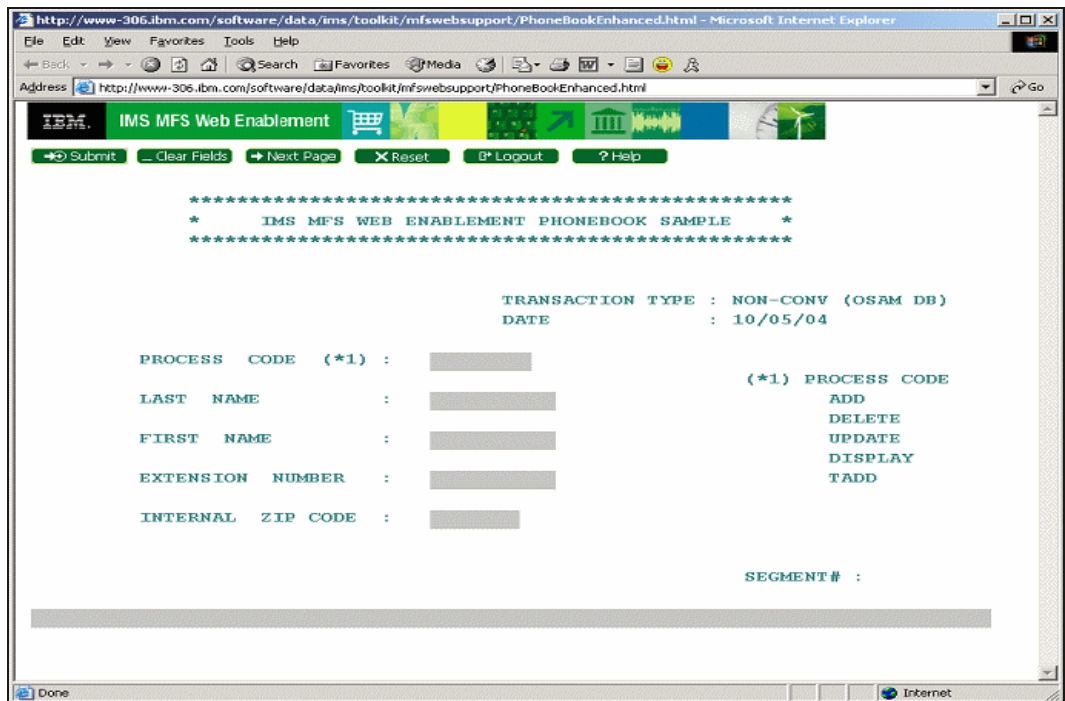


Figure 12-15 The formatted IMS IVP Phonebook panel obtained by issuing “/FOR IVTNO”

Transaction code followed by optional data

The transaction code and optional data are written to the input byte array and are then sent to IMS. The MFS Adapter converts the transaction code to uppercase text. The data remains unchanged (mixed cased allowed). The output execution follows the same flow as in the processing execution in “transaction data” from a Web page other than the initial page.

Figure 12-16 on page 259 displays the input of transaction code TTT3D on the initial blank panel.

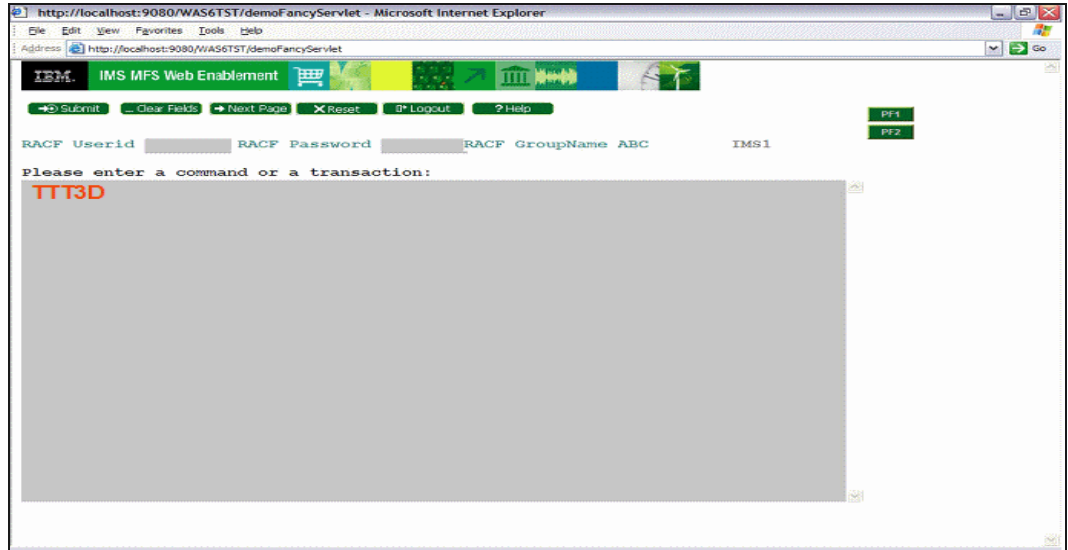


Figure 12-16 Input of transaction code TTT3D on the initial blank panel

Figure 12-17 displays the three dimensional tic-tac-toe format panel that is returned from the transaction code input of TTT3D.

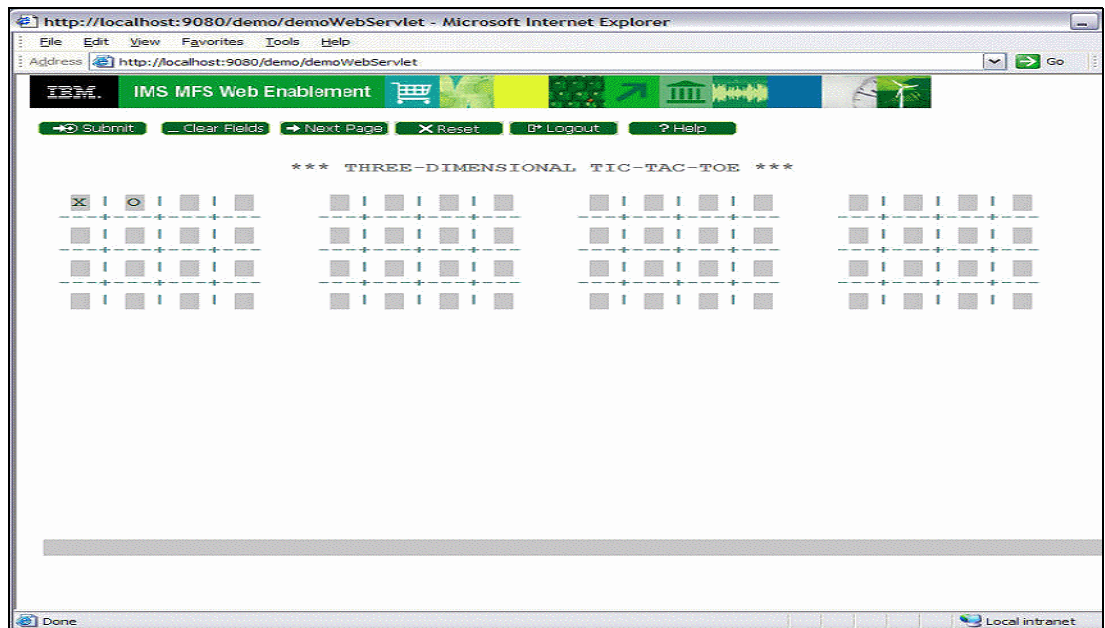


Figure 12-17 Output of the transaction TTT3D

The /EXIT command support

The MFS Servlet and the MFS Adapter process the `exit` command to end the current pending conversation. The MFS Servlet determines if the client is in the middle of a conversation. To end a conversational message, the MFS Adapter sets the `SYNC_END_CONVERSATION` in the `IMSInteractionSpec` and then sends an empty request through the IMS TM Resource Adapter to terminate the host conversation.

The system returns one of the following messages:

If in a conversation:

- ▶ DFS058I HH:MM:SS EXIT COMMAND COMPLETED.

If not in a conversation:

- ▶ DFS180 HH:MM:SS NO ACTIVE CONVERSATION IN PROCESS, CANNOT PROCESS COMMAND.

12.2.7 Connection Management

Figure 12-18 illustrates how the session connection is handled in MFS Web Enablement.

The MFS Servlet works with the HTTP session objects to:

- ▶ Load state information that is associated with the unique session ID.
- ▶ Create a new session if the request comes in with new session ID.
- ▶ Manage and update the state information in each HTTP request.
- ▶ Invalidate sessions when an HTTP session becomes unbound (upon logout, browser closed, or session time-out).

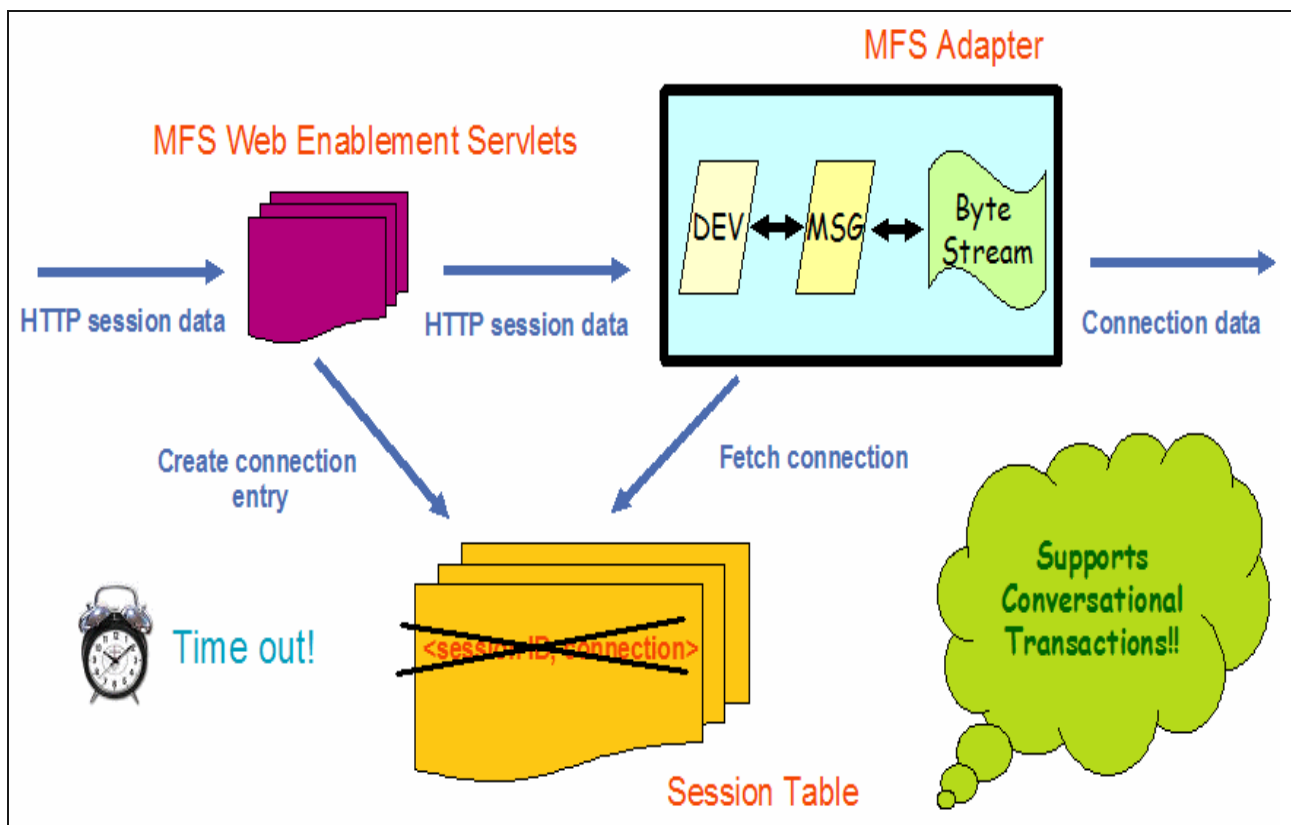


Figure 12-18 Session management controls

12.2.8 Supported core MFS features

The following features of core MFS are supported:

- ▶ 3270 type devices
- ▶ Attribute bytes
- ▶ Cursor positioning
- ▶ Extended attributes bytes (blinking only supported in Mozilla)
- ▶ Multiple physical pages input
- ▶ Multiple logical and physical pages output
- ▶ Message options 1 and 2 only for input and output
- ▶ PA1 key equivalent to advance to the next physical page
- ▶ PF keys with literal data (transaction code and two commands: /FOR and /EXIT) and two control functions: NEXTPP (next physical page) and ENDMPPPI (end multiple physical pages input)
- ▶ System literals only for date, time, and LPAGENO, System default MIDs and MODs, including DFSMI1, DFSMI2, DFSMO1, DFSMO2, DFSMO3, DFSMO5, and the blank panel.

Support for drop-down lists

Figure 12-19 displays how MFS Web Enablement can support modern Web browser technology, such as drop-down lists.

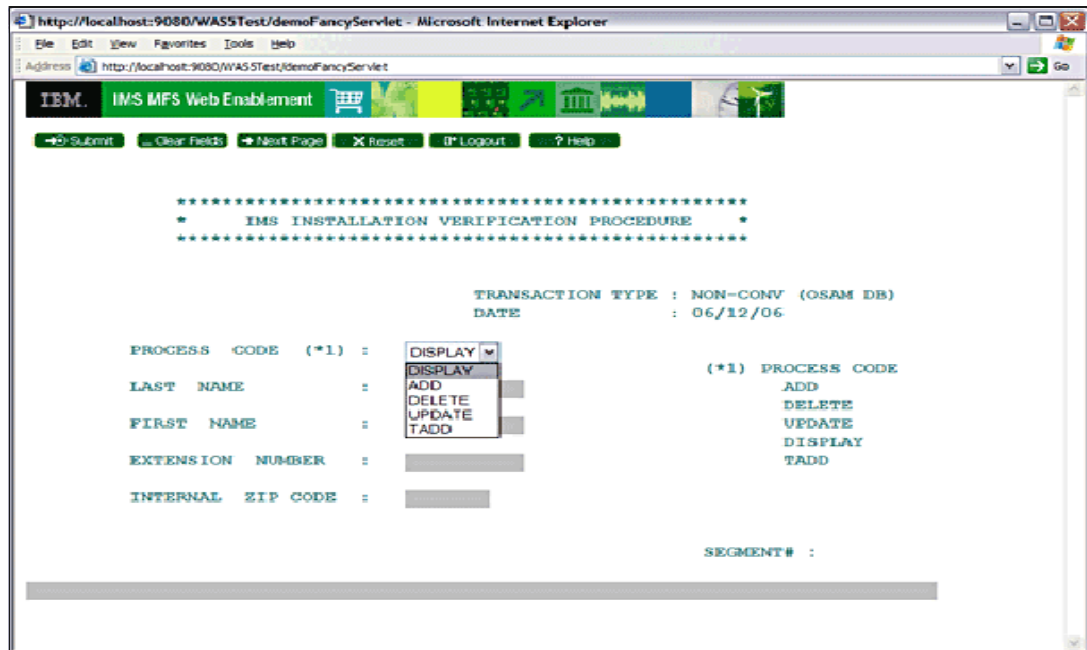


Figure 12-19 Drop down list support by MFS Web Enablement

Next, we discuss the other IMS Web Solution, which is MFS SOA support.

12.3 MFS SOA support

IMS Message Format Service SOA support is the next generation of MFS support and is integrated with the J2C (Java EE Connector Architecture) tools wizard in the IBM Rational Application Developer Version 7.5. Using IMS MFS SOA support, you can reuse existing MFS-based IMS business logic as Web services, Web pages, or Enterprise JavaBeans (EJBs). You can use the wizard to transform existing MFS-based IMS applications to J2C Java beans and J2C Java data binding classes. After you create a J2C Java bean, you can create Java EE resources, such as JavaService Pages (JSPs), EJBs, or Web services, to package the functions of the generated J2C Java bean. An application developer or application server administrator then deploys the EAR to run on a WebSphere Application Server V6.1.

During runtime, the deployed MFS service can be invoked in many ways to transform data to and back from IMS.

12.3.1 Why was MFS SOA Support developed

MFS SOA Support provides the flexibility and reusability that are needed to take advantage of existing MFS-based IMS transactions. Integration is the key here, and MFS SOA Support provides the support that is needed to integrate your MFS-based IMS transactions into your enterprise infrastructure.

J2C bean

At the center of the MFS SOA Support is the J2C Java bean.

Using the J2C tools, you can generate a J2C bean (consisting of an interface and implementation files) with one or more methods that correspond to functions on back-end systems. For IMS, the input and outputs to these functions are specified by separate data binding classes. After you create a J2C bean, you can then create Web pages, Enterprise JavaBeans, or a Web Service for your J2C bean.

Comparing MFS Web Services support in WSAD-IE

In this section, we outline the differences between the old MFS Web Services support that is provided through WebSphere Application Developer Integration Edition (WSAD-IE) V5.1.1 and the MFS SOA Support in Rational Application Developer V7.5.

WSIF versus J2C

MFS Web Services support in WebSphere Application Developer Integration Edition (WSAD-IE) uses Web Service Invocation Framework (WSIF) framework. In the core of the WSIF framework are the WSDL and XSD files, which contain:

- ▶ Metadata that represents interface and Enterprise Information System (EIS) operations
- ▶ Data format for input/output messages
- ▶ Client stub proxy beans (RPC: Remote procedure call) used to generate command proxy beans, format handler classes, and session beans

The J2C framework of Rational Application Developer (RAD) has the following features:

- ▶ J2C Java bean and data binding beans contains all the metadata information, which is then used to generate Java EE resources such as Web Services, Web page, and EJBs.
- ▶ JCA 1.0/1.5

Comparing tooling artifacts that are generated with WSADIE and RAD

Figure 12-20 presents the generated artifacts from WSADIE and RAD.

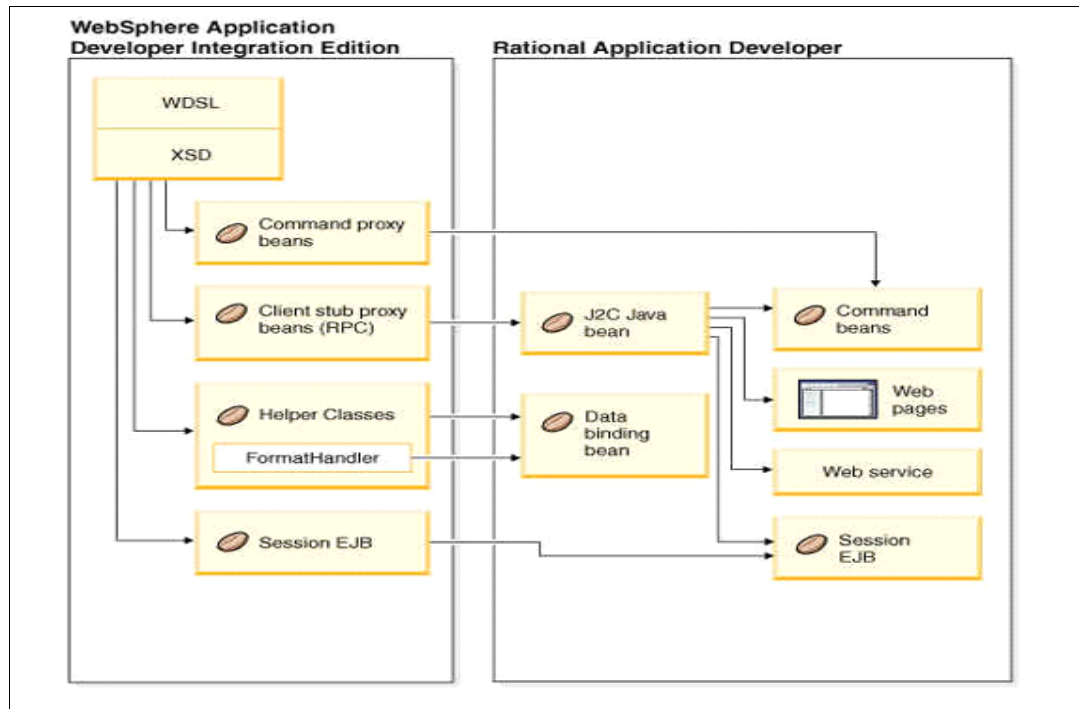


Figure 12-20 WSADIE and RAD generated artifacts

12.3.2 Development and runtime environments

MFS SOA requires a development environment that includes the following software:

- ▶ Rational Application Developer V7.5.
- ▶ IMS TM Resource Adapter V9.1.0.2.5a or V10.2.0. and beyond

The runtime environment consists of the following software:

- ▶ IMS Version V9.1 and beyond
- ▶ IMS Connect V9.1 and beyond
- ▶ IMS TM Resource Adapter V9.1.0.2.5a, V10.2.0 and beyond
- ▶ WebSphere Application Server V6.1

IMS TM Resource Adapter

MFS SOA support is delivered through IMS TM Resource Adapter. When IMS TM Resource Adapter with MFS SOA support is installed in Rational Application Developer V7.5, the MFS SOA EMD (Enterprise Metadata Discovery) wizard can be accessed from the J2C dynamic wizard.

Note: Not all IMS TM Resource Adapter versions come with MFS SOA support. At the time of the creation of this book, MFS SOA support is available in IMS TM RA version 91025a and 1020.

Rational Application Developer Version 7.5 tooling

MFS SOA Support only works with Rational Application Developer V7.5 and does not work with other Rational Application Developer versions, even if IMS TM Resource Adapter with

MFS SOA support is installed without problems. RAD 7.5 comes with the latest support for the MFS based conversational application.

12.3.3 Functional components of MFS SOA solutions

MFS SOA includes a development component that lets you create your application using an IDE (Integrated Development Environment), such as IBM Rational Application Developer. The runtime component needs to be deployed on WebSphere Application Server V 6.1 and access your IMS through IMS Connect and the IMS TM Resource Adapter. Figure 12-21 displays a high-level schematic-like view of developing and deploying a J2C-based service or Web Service using Rational Application Developer.

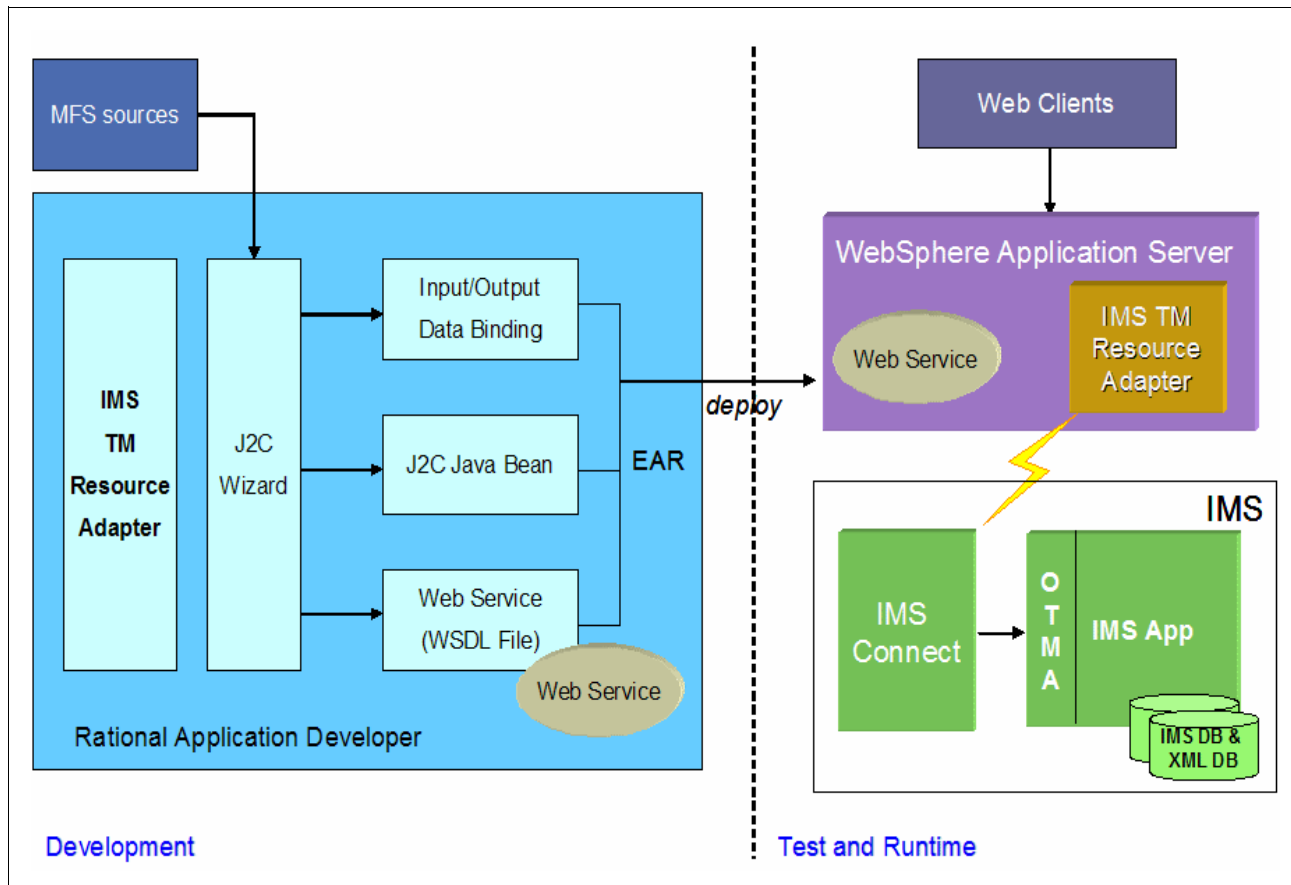


Figure 12-21 Using Rational Application Developer to develop and deploy a service

MFS SOA EMD wizard

The MFS SOA EMD wizard, integrated into the J2C wizards in RAD V 7.5, invokes the MFS SOA Importer to parse MFS source files and creates services for MFS-based IMS transactions. The MFS SOA EMD wizard guides application developers through the selection of MFS Message Input Descriptors (MID) and Message Output Descriptors (MOD) and populating the service definition, including the XML schema for input/output types. The service description contains all of the information that is required for the MFS data binding generator to generate MFS J2C Java Data Binding files.

MFS SOA Importer

The MFS SOA Importer, integrated in the MFS SOA EMD wizard, uses Eclipse Modeling Framework (EMF) to transform MFS source into EMF resources from which XSD can be generated.

MFS EMF resources capture all of the relationships and properties of the MFS source file.

MFS input/output data bindings

Data bindings provide a Java application with methods for populating the input message with data and for retrieving data from the output message. Data bindings also handle platform-related functions, such as the conversion between UNICODE and EBCDIC.

J2C Java bean

After input/output data bindings are created, we must create a Java bean that communicates with IMS through the Java EE Connector Architecture.

This Java bean includes a method that submits a request to IMS to run the IMS transaction. This method uses the data bindings to build the input and output messages for the transaction. A J2C Java bean can include more than one method that runs an IMS transaction and multiple data bindings for different input and output messages. The code that is generated for the J2C Java bean uses the CCI provided by the IMS resource adapter to communicate with IMS.

Managed and non-managed connections

Managed connections are created by a construct of the Java EE Connector Architecture called a connection factory and are managed by the application server. Your Java bean accesses a connection factory using Java Naming and Directory Interface (JNDI). Managed connections are recommended. The IMS resource adapter and the application server's connection manager work together to efficiently manage connections by providing connection pooling, reuse, and persistence.

Non-managed connections are obtained directly through the IMS resource adapter, without collaboration with the application server. Non-managed connections are typically used by two-tiered applications and are not pooled or reused. In addition, non-managed socket connections between the IMS resource adapter and IMS Connect are not persistent, incurring the additional overhead of opening and closing the socket for each use by an application.

12.3.4 Creating Java EE resources and embedding the J2C bean

After the J2C Java bean and the data bindings are created, the J2C bean can be embedded into Java EE resources, such as Web page, Web Service, or EJB, using the J2C wizard that Rational Application Developer provides.

Figure 12-22 on page 266 illustrates the wide range of possibilities with the generated J2C Java bean.

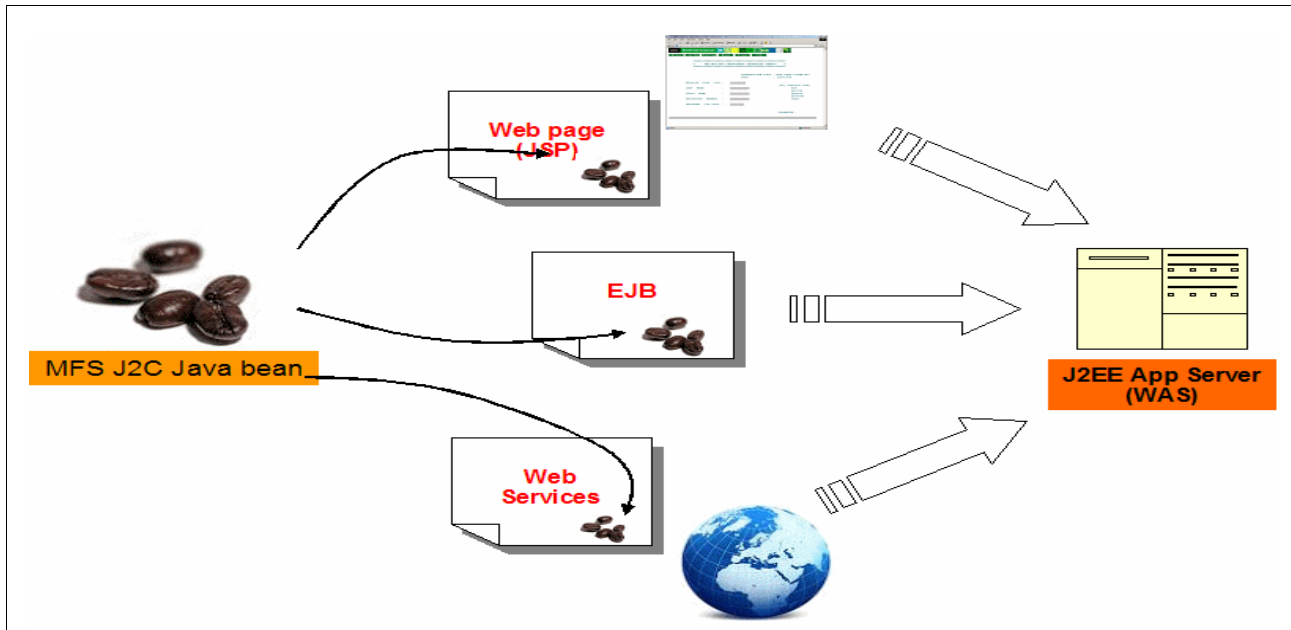


Figure 12-22 Generated J2C Java bean uses

Figure 12-23 presents the selection of the wizard to generate Java EE resources from the previously created J2C Java bean.

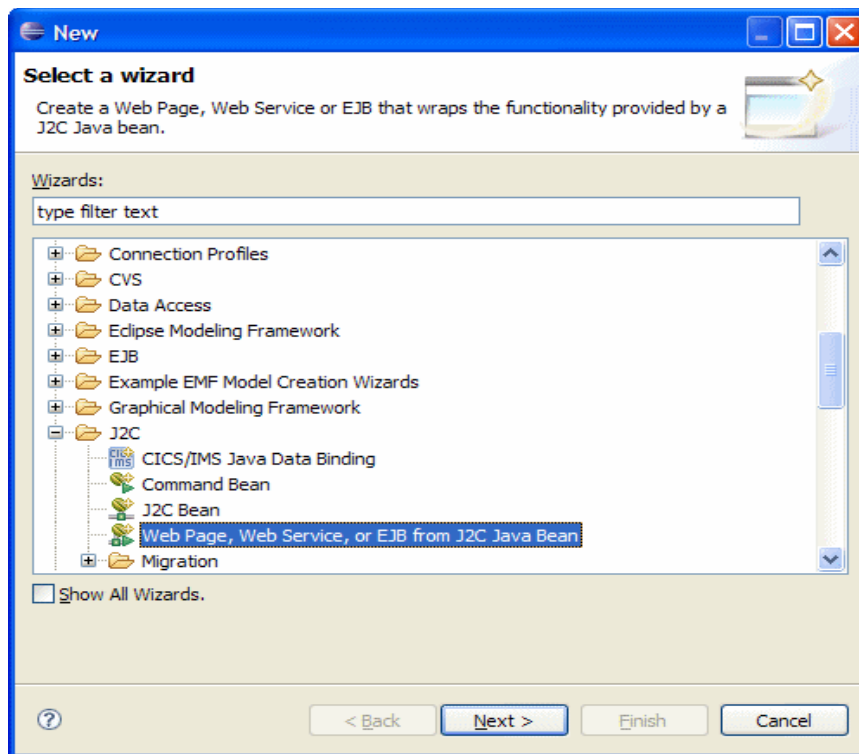


Figure 12-23 Wizard to generate Java EE resources

Figure 12-24 on page 267 shows the type of resources that you can generate to embed the J2C Java bean.

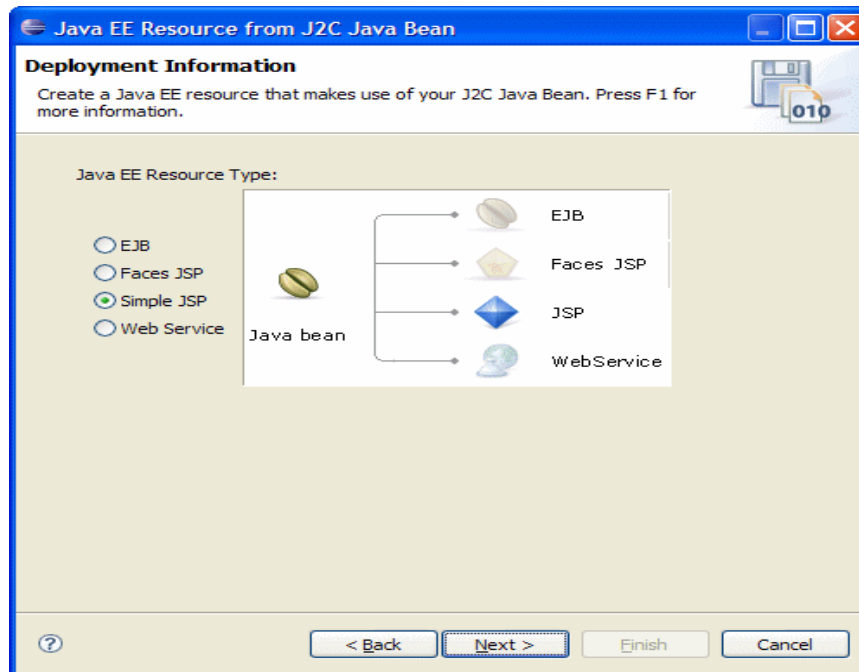


Figure 12-24 Types of resources that you can generate

12.3.5 Conversational support

In IMS Version 9, IMS and IMS Connect track the conversation. In IMS Version 10, a new parameter, ConvID, is introduced to track the conversation. The ConvID parameter is the conversational ID that is passed between IMS Connect and IMS. The MFS SOA client is responsible for keeping track of the ConvID parameter in the iterations after the initial send.

Conversational processing in IMS Version 9

The flow for conversational processing in IMS Version 9 is:

1. The MFS client sends the initial iteration of a conversation with the transaction code and data through IBM WebSphere Application Server to IMS Connect.
2. IMS Connect sends the message to IMS and keeps track of a token that it uses for the subsequent iterations.
3. IMS sends the response to the MFS client. In the subsequent send from the MFS client, IMS Connect knows by the client ID and port number and the token that this client is in a conversation and keeps the conversation active.
4. The MFS client sends a message indicating that it wants to end the conversation, and the conversation is terminated.

Non-managed connection scenario: MFS SOA Conversational application for IMS V9 currently only supports non-managed connection scenario.

Conversational processing in IMS Version 10 and later

The flow for conversational processing in IMS Version 10 and later is:

1. The MFS client sends the initial iteration of a conversation with the transaction code and data through IBM WebSphere Application Server to IMS Connect.
2. IMS Connect then sends the message to IMS.
3. IMS returns a response and a conversational ID to the MFS client, which retrieves the ID from the output, and on the subsequent reply supplies the ID in the input. This way, IMS retrieves the conversational ID and keeps the conversation active.
4. The MFS client sends a message indicating that it wants to end the conversation, and the conversation is terminated.

Exposing parameters for conversational transaction

You can expose parameters that are available for IMS V10, such as convEnded, convID, and useConvID from InteractionSpec, for input and output using Rational Application Developer V7.5.

After these parameters are exposed, the generated J2C bean has access to them and the generated conversational wrapper.

Figure 12-25 on page 269 presents the panel that is used to expose the input or output parameters.

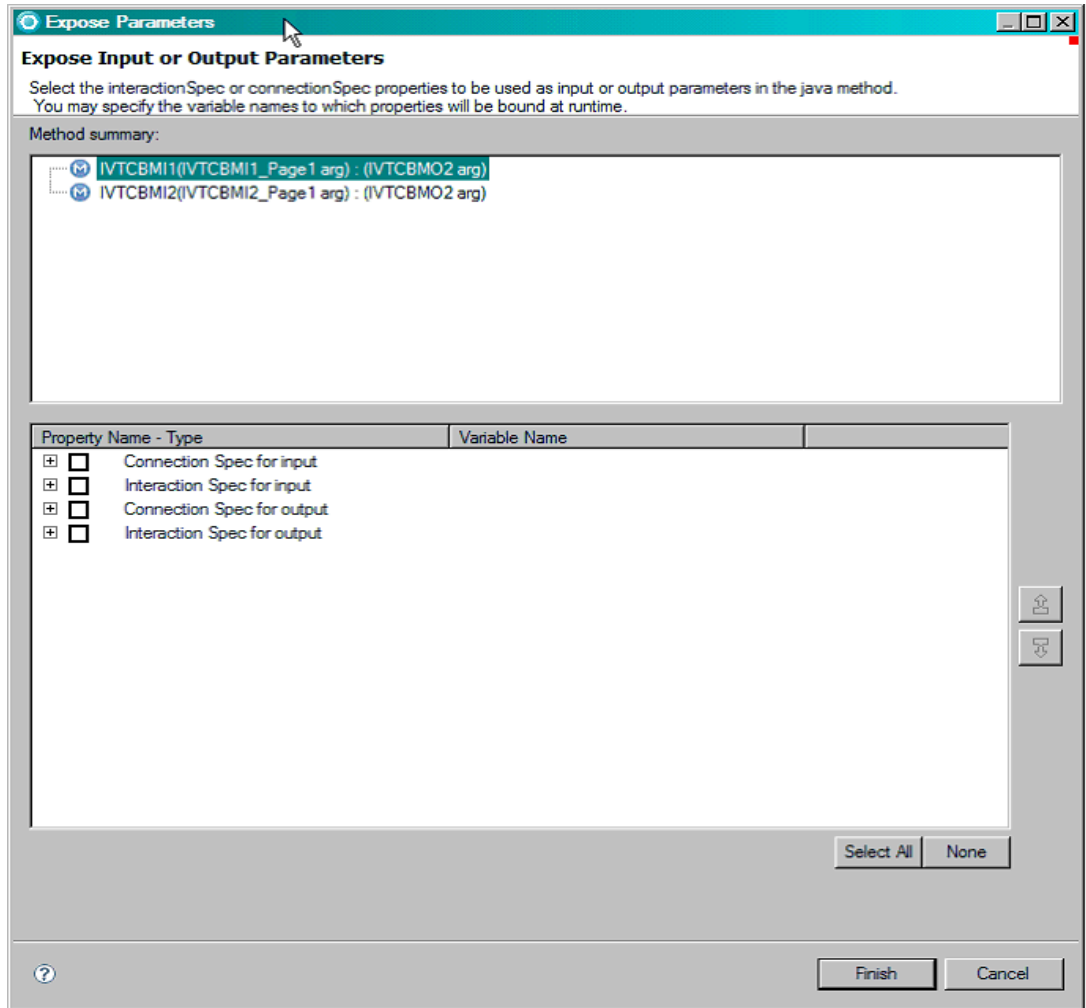


Figure 12-25 Exposing input or output parameters

Figure 12-26 on page 270 displays the selection of the interactionSpec or connectionSpec properties.

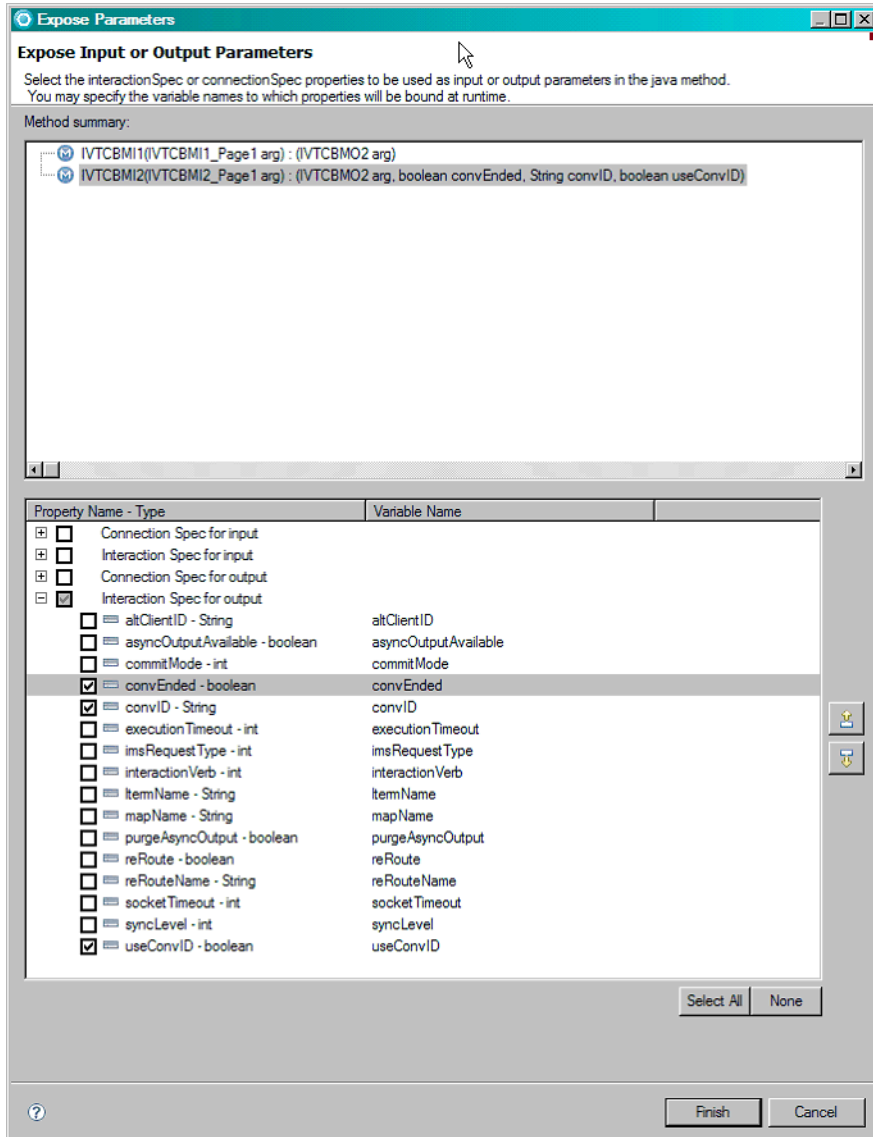


Figure 12-26 Selecting the InteractionSpec properties

This concludes our examination of the MFS in SOA modernization.



IMS Web 2.0 Solution

The IMS Web 2.0 Solution provides business value for customers by providing a quick way to build rich internet applications through Web 2.0 technologies. IMS transactions can be exposed as Web 2.0 RESTful (Representational State Transfer) services, which then can be consumed by other Web 2.0 tools to quickly create a Web mashup that relies on enterprise data from different feed sources.

IMS Web 2.0 Solutions is embedded in IBM InfoSphere MashupHub to transform IMS assets into Atom feeds.

With IMS Web 2.0 Solution, you can create Atom feeds in MashupHub from IMS transactions that run on IMS Version 10, with integrated IMS Connect Version 10. Feeds can be further restructured and customized by using the operators and functions in MashupHub. Web-savvy business users can then use IBM Lotus® Mashups to mash information from various sources into a single view of information, and create a flexible, dynamic application.

The terms RESTful services, MashupHub, and Atom feeds might be confusing to you at the moment, but we explain them in this chapter.

13.1 What is Web 2.0

The term Web 2.0 refers to a new trend of user-driven internet activity, including information sharing, networking, collaboration and user-driven content. These concepts led to the development of Web-based communities, such as social-networking sites, wikis, blogs, and so on. Web 2.0 encompasses both new styles of using the Web and the standards to support it. Web 2.0 differs from the original WWW in the manner that software developers and end-users use the Web.

Web 2.0 empowers users to manipulate data and combine various services into a single Web experience. It also enables businesses to gain a competitive edge when they put the vast amounts of data they own about themselves and their customers to creative uses.

Many of the elements of Web 2.0 are highly relevant to SOA and the service economy. We can identify four characteristics in particular:

- ▶ Loosely coupled systems:
 - Small pieces loosely joined – decoupling data from functionality over the internet – sometimes called “web as platform “
 - Granular addressability of content
 - Radical decentralization
- ▶ Remixability:
 - Repurposing services and other assets
 - Unlocking valuable business data
- ▶ User-centric:
 - Users control their own data
 - Rich user experience, and Rich Internet Applications (RIA)
 - User behavior not predetermined
 - Customer self-service based
- ▶ Enterprise social software:
 - Networking and ad hoc computing within the enterprise.

13.1.1 Web 2.0 and IMS

Web Oriented Architecture (WOA) is leading the way to the future.

WOA is an instance of SOA that uses concepts from the Web as the primary service architecture, whereas the Web is the platform for SOA. Therefore, WOA is a simplified implementation of SOA, and evolved to become the preferred choice of Web-facing systems. Although traditional SOA encompasses transforming reusable business tasks, such as IMS assets into services, WOA is a constrained subset of SOA that is dedicated to making those services easily and readily available to the Web community.

Web 2.0, based on WOA, is a major implementation that is emerging for Web-based applications. Many people who access the Web want to customize their blogs, Web sites, or both with content and feeds from elsewhere on the Web. Web 2.0 empowers Web users to be both consumers and producers by manipulating data and combining various sources into a single Web experience.

With Web 2.0, a massive population of Web users is enabled to develop Web mashups and to contribute to the ever-evolving Internet like never before. In a business setting, Web 2.0

allows companies and corporations to gain a competitive edge by leveraging the content that they own to open up new business uses and opportunities.

IMS Web 2.0 Solution aims to unleash enterprise assets that run crucial business processes. As part of Info 2.0; IBM's Web 2.0 initiative, IMS Web 2.0 simplifies the integration of IMS data and content, allowing for shorter deployment cycles. IMS Web 2.0 extends the usability of IMS assets by adding the ability to remix and mashup data. IMS Web 2.0 also provides the ability to compose and build services and widgets into composed services and user interfaces. Using IMS Web 2.0, IMS customers can bring existing IMS transactions onto the Web more effectively and often at the same or lesser cost.

Figure 13-1 shows the power of content mashup in Web 2.0

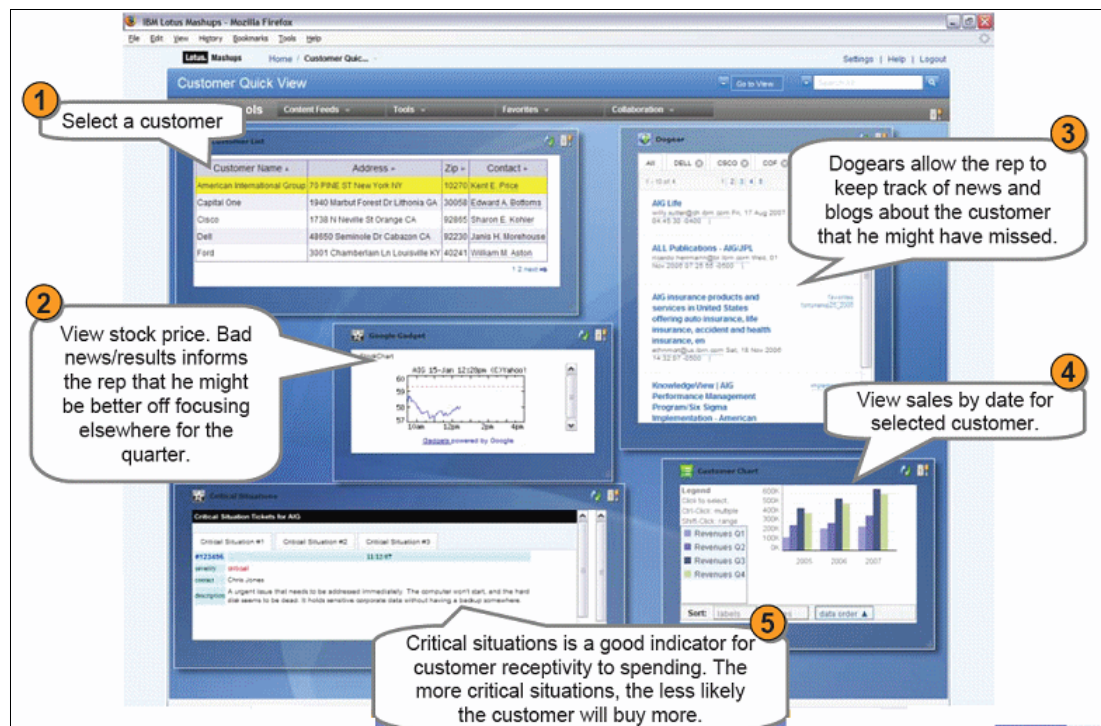


Figure 13-1 Web 2.0 content mashup

13.1.2 Business benefits

One of the important benefits of Web 2.0 is that prior to Web 2.0 application development was within the domain of IT department. With Web 2.0 non-technical people such as business analysts can create new applications. Here is an example of how dynamically this new technology can be utilized.

Figure 13-2 on page 274 shows a sample MFS based IMS transaction screen through Web browser (refer to 12.2, “MFS Web Enablement” on page 249 for information on how this was created). With the power of IMS MFS Web Solutions, IMS transaction data now can be accessed through a Web browser and TCP/IP, instead of through 3270 terminals or VTAM. However, the data itself is still one dimensional. Data from IMS is shown as pure data without augmenting the user experience. When IMS data is used in this way, no additional synergy is created along side the data.

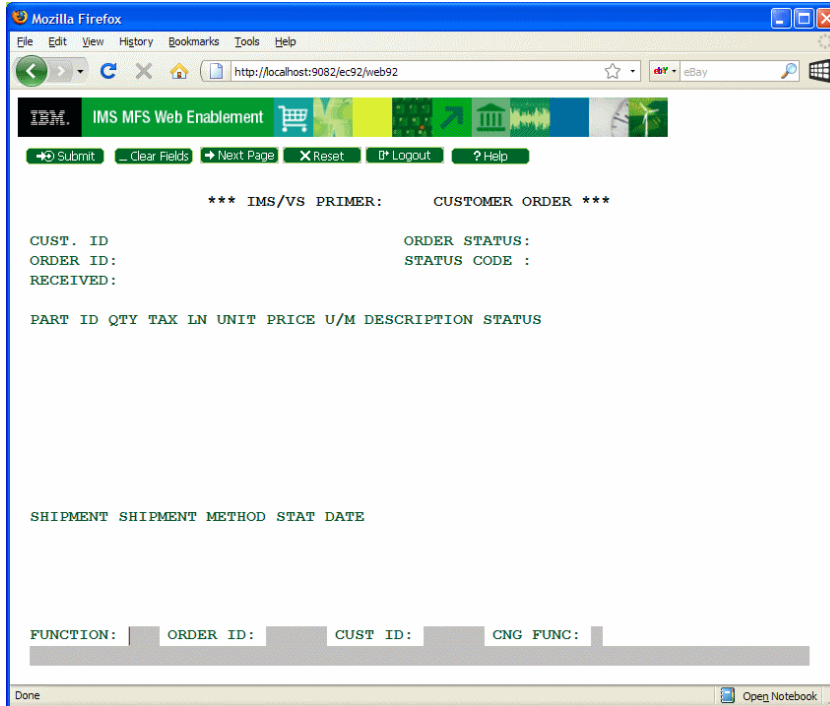


Figure 13-2 Sample MFS based IMS transaction screen through a Web browser display

In this example, we place ourselves in the position of an automotive manufacturing executive placed with challenging decisions to make.

Figure 13-3 on page 275 and Figure 13-4 on page 276 show mashups composed of feeds created from IMS transactions, IMS databases, Excel® Spreadsheets, and additional feeds imported from WebSphere sMash and the Internet.

In both diagrams on the top left, there is a list of settlement reports. This table is populated by a feed pulled from querying the IMS settlement database, and each entry lists the failed part, and associated settlement cost.

Also in both diagrams on the top right, we see the locations on a Google Map, showing all the related airbags accidents in 2007.

In Figure 13-3 on page 275 on the bottom left, we have the recall analysis for airbags, using the recall cost feed we aggregated earlier and a feed from a spreadsheet about estimated total settlement costs. In this case, decision makers can see that making a recall on the airbags is justified by the cost analysis.

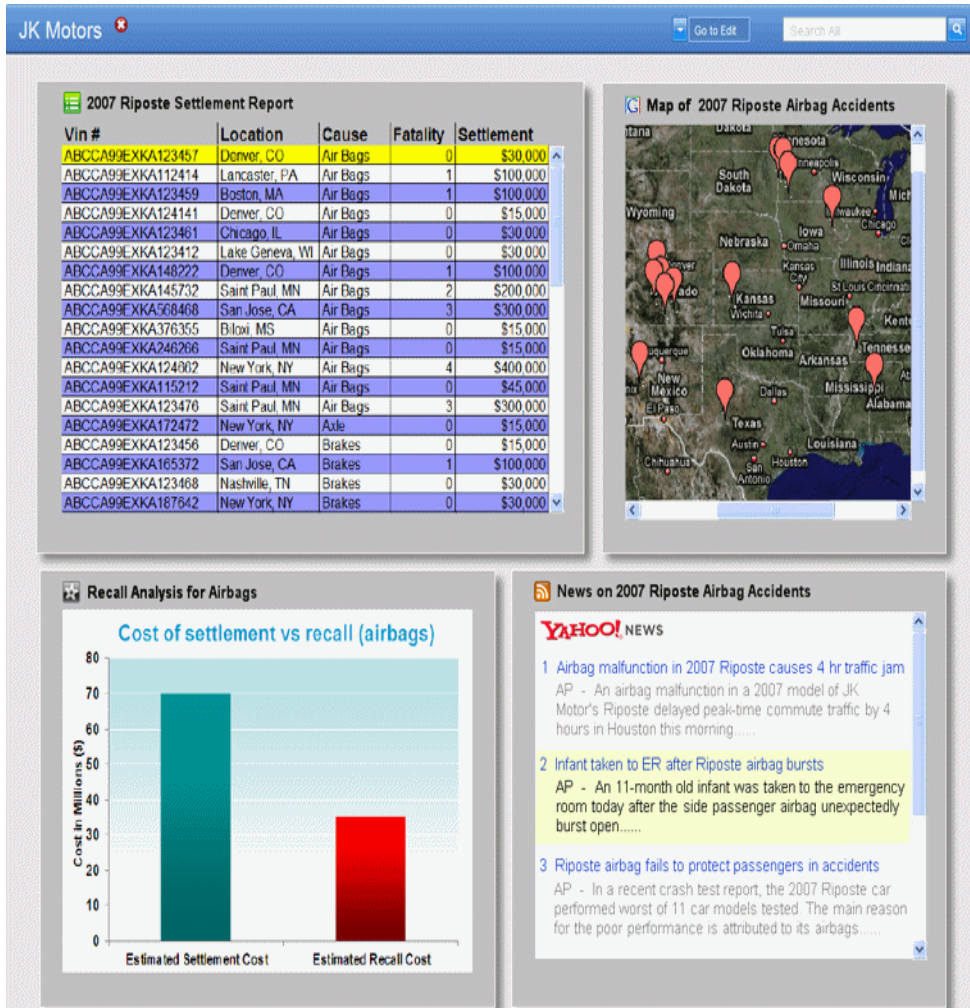


Figure 13-3 A mashup composed of feeds created from various sources

In Figure 13-4 on page 276 on the bottom left, a bar chart from settlement to recall costs for brakes is presented for review.

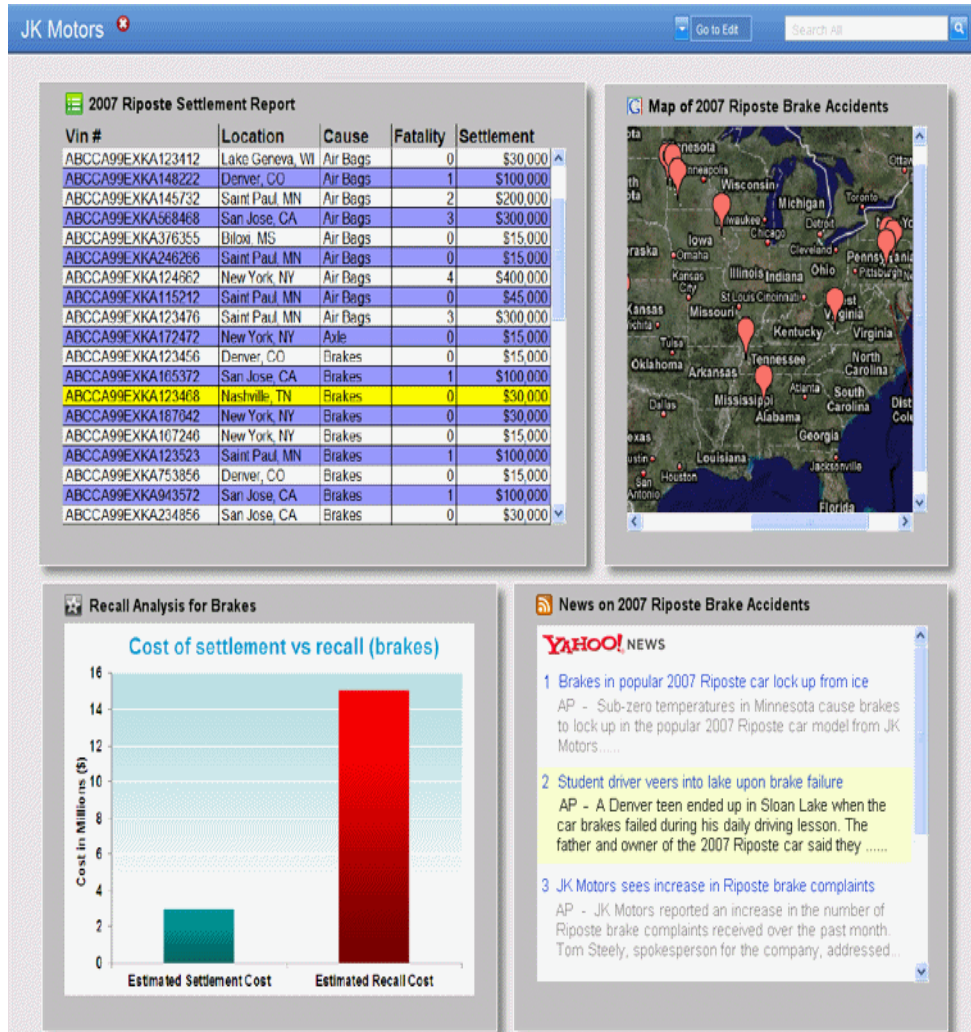


Figure 13-4 Another mashup composed of feeds created from various sources

In both diagrams, on the bottom right, a feed showing news clips to allow us to better evaluate the decision and how it affects our company's public image.

Now we have more than just data, but valuable information that can be used by people to make decisions.

13.1.3 Web 2.0 and services-oriented architecture

How do we connect people and systems together easily? How do we make software and data available for reuse through services and how can we build new value on existing information resources and IT assets? All these questions can be answered by the convergence of Service Oriented Architecture and Web 2.0.

Figure 13-5 on page 277 highlights the differences between SOA and Web 2.0.

Context	Provide	Remix	Assembly
SOA offers	Delivery of information and system solutions to internal and external users in the form of services	Availability of data to remix into new applications/services Loosely coupled abstractions to support repurposing and remixing	Consumption of third party information and system solutions (e.g. packaged applications) in the form of services
Web 2.0 offers	Extra sources of information and services Management and rendering of user content	Light-weight access protocols for SOA services	Extra channels for delivering enterprise content to consumers Channels for peer-to-peer collaboration between users

Figure 13-5 Comparison between Web 2.0 and SOA

SOA and Web 2.0 convergence is a hot topic because many of the biggest challenges in enterprise IT are being solved today out on the Web domain, particularly in the following areas:

- ▶ Engaging users
- ▶ Delivering highly usable software
- ▶ System integration
- ▶ Reusability

Figure 13-6 shows how the convergence of Web 2.0 and SOA help to represent data in the enterprise world from a new perspective.

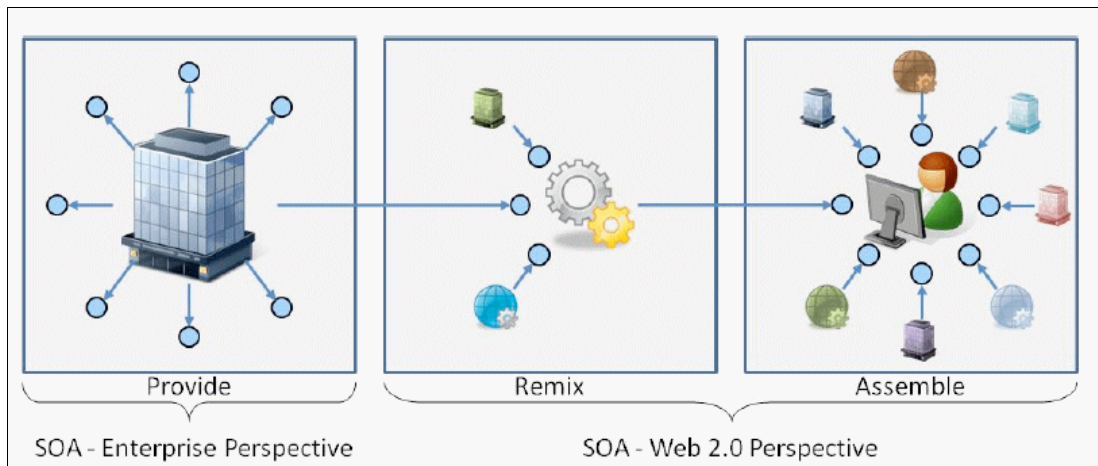


Figure 13-6 The convergence of Web 2.0 and SOA roles

Figure 13-7 on page 278 helps describe Figure 13-6 by highlighting the roles of SOA and SOA with Web 2.0.

Provide	○ SOA is designed to make information more easily accessible via services, and to a wider audience both within and outside the enterprise
Remix	○ Web 2.0 helps add value by combining enterprise services and web feeds together to help deliver richer information
Assemble	○ Web 2.0 is designed to make it easier to assemble interactive solutions using services from many different sources

Figure 13-7 Roles that SOA and Web 2.0 perform

13.2 Definition of key terminology

As we mentioned in the preface of this chapter, it is important to understand the new terminology associated with Web 2.0.

13.2.1 What is a REST service

The term REST describes a design pattern for implementing networked systems. REST is neither a technology nor a standard; it's an architectural style for exposing resources over the Web. Clients and servers adhere to a uniform interface. All resources are accessed with a generic interface in the Web-extended SOA world by HTTP methods of: GET, POST, PUT, and DELETE. Clients access named resources. The system comprises resources that are named using a URL, such as an HTTP URL. Figure 13-8 presents this concept.

REST

- **Representational State Transfer – REST publishing protocol**
 - Emerging style for interacting with web services / ATOM feeds →
 - Involves sending a request to a special URL and receiving an XML document containing the server's response
 - Verbs are a subset of the verbs in the HTTP protocol


```
GET http://myserver/partsdepot/0035.img
PUT http://myserver/partsdepot/part123
DELETE http://myserver/partsdepot/oldstuff.img
POST http://myserver/myblog
```
 - Everything is a resource accessible with a URI

REST	DB access	Cut and Paste
POST	CREATE (Insert)	PASTE AFTER
GET	READ (Get)	COPY
PUT	UPDATE (Replace)	PASTE OVER
DELETE	DELETE (Delete)	CUT

API comparisons

Figure 13-8 Use of HTTP protocol verbs for REST

Requests are client-server based, and by nature use a pull-based interaction style. Consuming components pull representations of state from the server. Requests are stateless.

Each request from client to server must contain all the information needed to understand the request and cannot take advantage of any stored context on the server.

REST is a key technology for representing services on the Web.

REST services versus SOAP based Web Services

In this section, we compare REST and SOAP-based Web Services:

- ▶ REST is resource oriented, while Web Services (WS) are service oriented.
- ▶ REST mostly uses HTTP as the transport, and although SOAP normally uses HTTP as the transport mechanism, it has no restriction to bind it to a particular transport.

Both models have their strong and weak points, and sometimes they become complementary. For more information on RESTful Web services, refer to the Web site:

<http://java.sun.com/developer/technicalArticles/WebServices/restful/>

When to use REST

Architects and developers need to decide when this particular style is an appropriate choice for their applications. A RESTful design is appropriate when:

- ▶ The Web services are completely stateless. A good test is to consider whether the interaction can survive a restart of the server.
- ▶ A caching infrastructure can be leveraged for performance. If the data that the Web service returns is not dynamically generated and can be cached, then the caching infrastructure that Web servers and other intermediaries inherently provide can be leveraged to improve performance. However, the developer must take care because such caches are limited to the HTTP GET method for most servers.
- ▶ The service producer and service consumer have a mutual understanding of the context and content being passed along. Because there is no formal way to describe the Web services interface, both parties must agree on the schemas that describe the data being exchanged and on ways to process it meaningfully.
- ▶ Bandwidth is particularly important and needs to be limited. REST is particularly useful for limited-profile devices such as PDAs and mobile phones, for which the overhead of headers and additional layers of SOAP elements on the XML payload must be restricted.

When to use SOAP

A SOAP-based design can be appropriate when:

- ▶ A formal contract must be established to describe the interface that the Web service offers. The Web Services Description Language (WSDL) describes the details such as messages, operations, bindings, and location of the Web service.
- ▶ The architecture must address complex nonfunctional requirements. Many Web services specifications address such requirements and establish a common vocabulary for them. Examples include transactions, security, addressing, and coordination, and so on.
- ▶ The architecture needs to handle asynchronous processing and invocation.

13.2.2 What is a feed

A *feed* is a Web document (often XML-based) which provides users with frequently updated content. These documents contain lists of related information composed of a number of items, known as "entries", each with an extensible set of attached metadata. They are intended to support synchronization between publishers and consumers.


Feeds are consumed directly by users with aggregators or feed readers, which combine the contents of multiple Web feeds for display on a single screen or series of screens. There are different types of feeds. Among them Rich Site Summary (RSS) and ATOM are the most popular ones.


RSS is a family of Web feed formats used to publish frequently updated works such as blog entries, news headlines, audio, and video in a standardized format. An RSS document includes full or summarized text plus metadata, such as publish dates and authors.

The Atom format was developed as an alternative to RSS to clarify ambiguities, consolidate RSS's multiple versions, and expand its capabilities. Figure 13-9 summarizes the definitions of RSS and ATOM feeds.

XML, RSS and ATOM Feeds

- **Addresses the continued goal of simplification**
 - Feeds are XML documents containing lists of related information composed of a number of items, known as "entries", each with an extensible set of attached metadata (<http://www.ietf.org/rfc/rfc4287.txt>)
 - Web feeds allow programs to check for updates published on a web site

 **Atom** is an XML-based Web content and metadata syndication format, and an application-level protocol for publishing and editing Web resources (AtomEnabled.ORG)

 **RSS** is a web feed formats used to publish frequently updated works, e.g. blog entries, news headlines, audio, and video – in a standardized format

- **Feeds and REST**
 - POST creates XML, RSS, Atom entries and media files
 - GET retrieves the entry point Introspection document, collections (represented as feeds), and individual resources (entries or media files)
 - PUT updates entries and media files
 - DELETE removes an entry or a media file

Figure 13-9 XML, RSS, and ATOM feeds

13.2.3 What is a widget

A *widget* is a small program or piece of dynamic content that can be easily placed into a Web site. "Mashable" widgets pass events, so that they can be wired together to create something new.

13.2.4 What is a mashup

A *mashup* is a lightweight Web application created by combining information or capabilities from more than one existing source to deliver new functions and insights.

Mashup can be characterized by the following two properties:

- ▶ "Widgets" and "feeds" that are mashed together often come from independent sources and do not change when mashed.
- ▶ It is built on a Web-oriented architecture (REST, HTTP) and leverages lightweight, simple integration techniques such as AJAX, RSS, or JSON.

The result is fast creation of rich, desktop-like Web applications.

Figure 13-10 illustrates that the mashupable content can come from diverse feeds.

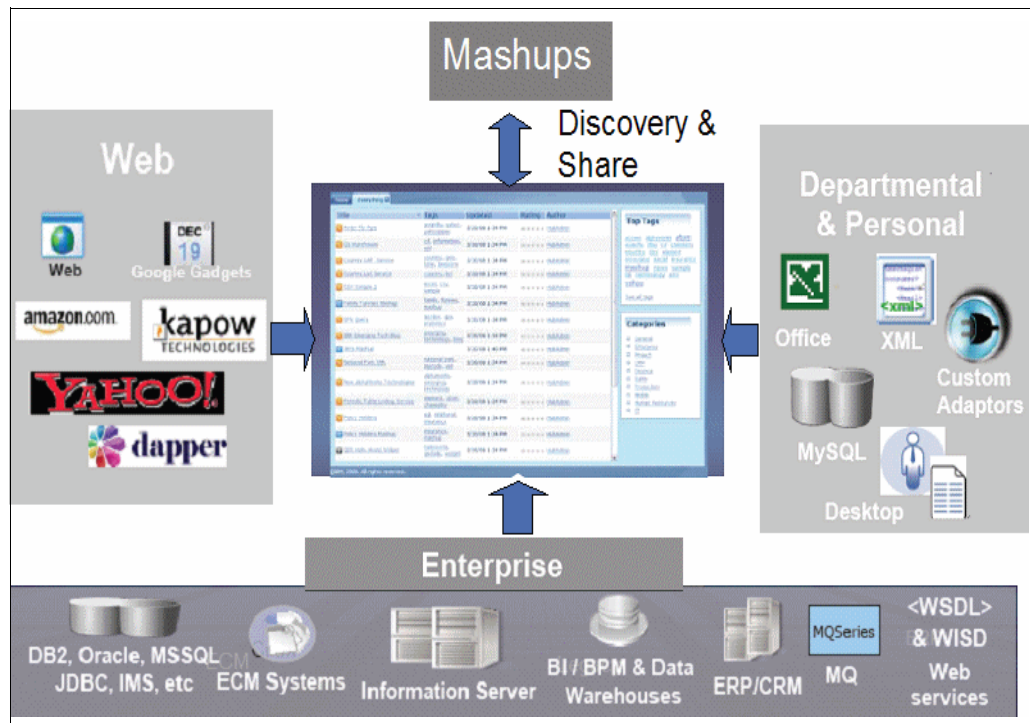


Figure 13-10 Mashupable content sources

A good example of a simple consumer mashup is the use of cartographic data from Google Maps to add location information to real-estate data, thereby creating a new and distinct web service that was not originally provided by either source.

13.3 IBM Mashup Center Enterprise Edition Version 1.1

IBM Mashup Center Enterprise Edition Version 1.1 includes InfoSphere MashupHub and Lotus Mashups.

With the IBM Mashup Center Enterprise Edition Version 1.1 you can integrate your existing IMS transactions with COBOL or PL/I source into Web 2.0 mashup and application solutions. These mashups or applications can consume and be consumed by other services such as XML, Atom, or RSS feeds.

Figure 13-11 on page 282 presents the IBM Mashup Center and the offerings that fall beneath it.

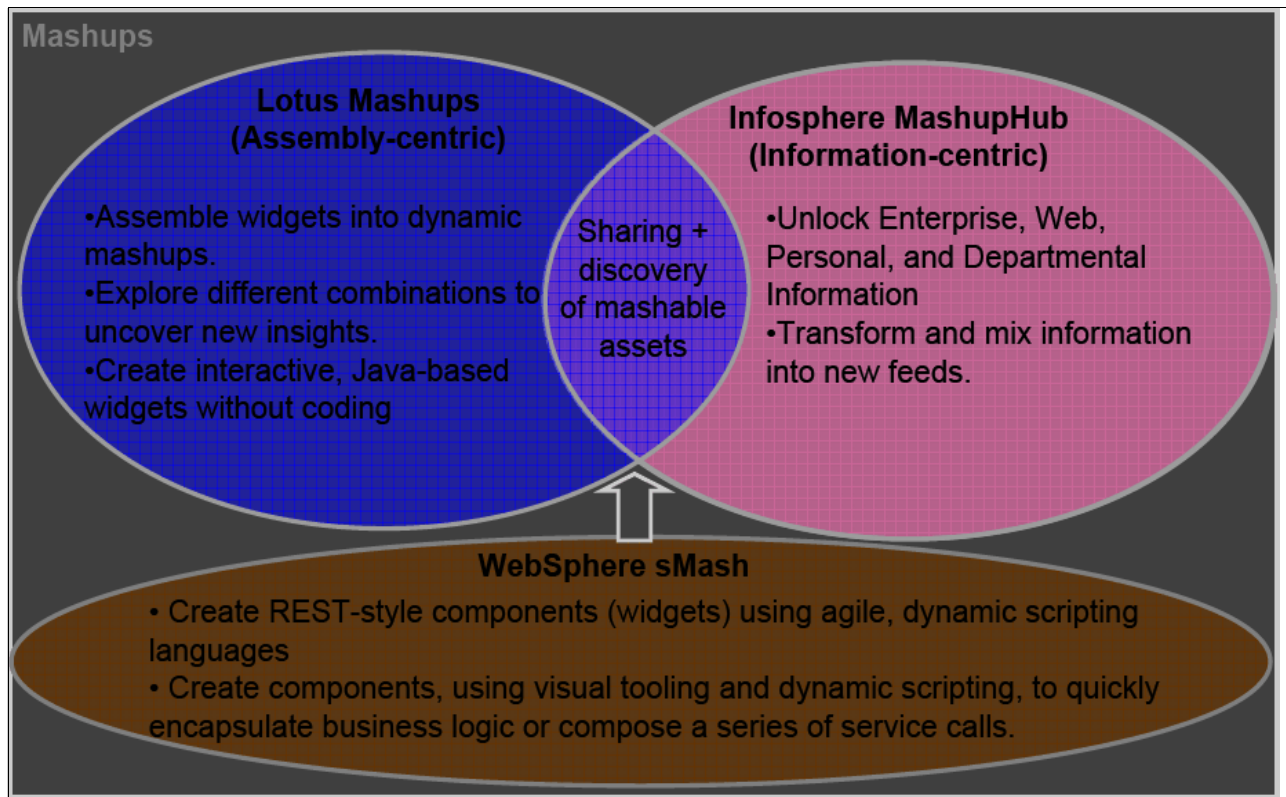


Figure 13-11 IBM Mashup offerings and functions

13.3.1 InfoSphere MashupHub

IBM InfoSphere MashupHub is a browser based tool for creating, storing, transforming, and remixing feeds. MashupHub also has a central catalog for users to tag, rate, and share mashable assets.

When you create feeds in MashupHub from data sources such as IMS applications, the metadata for the feed is added to the MashupHub catalog. When you view or use the feed, MashupHub generates the feed with the latest information from the data source. The feeds are displayed in the Atom feed format.

You can then use products, such as IBM Lotus Mashups, to enable Web-savvy business users to mash information from various sources— personal, enterprise, and Web content— into a single view of disparate sets of information, and create a flexible, and dynamic application.

13.3.2 Lotus Mashup

After assets are accessed and prepared, they can be assembled in the third step into a mashup with Lotus Mashup. Lotus Mashup is a browser-based tool that supports the assembly of situational applications by non-technical users. Figure 13-12 on page 283 presents an visual overview of the highlights of this offering.

Lotus Mashups: *Create Dynamic Widgets*

Easy-to-use, Eclipse-based IDE helps to reduce the time and cost of creating dynamic, interactive widgets. Using the tool's wizard-based interface, developers of all skill levels can build powerful widgets– without coding!

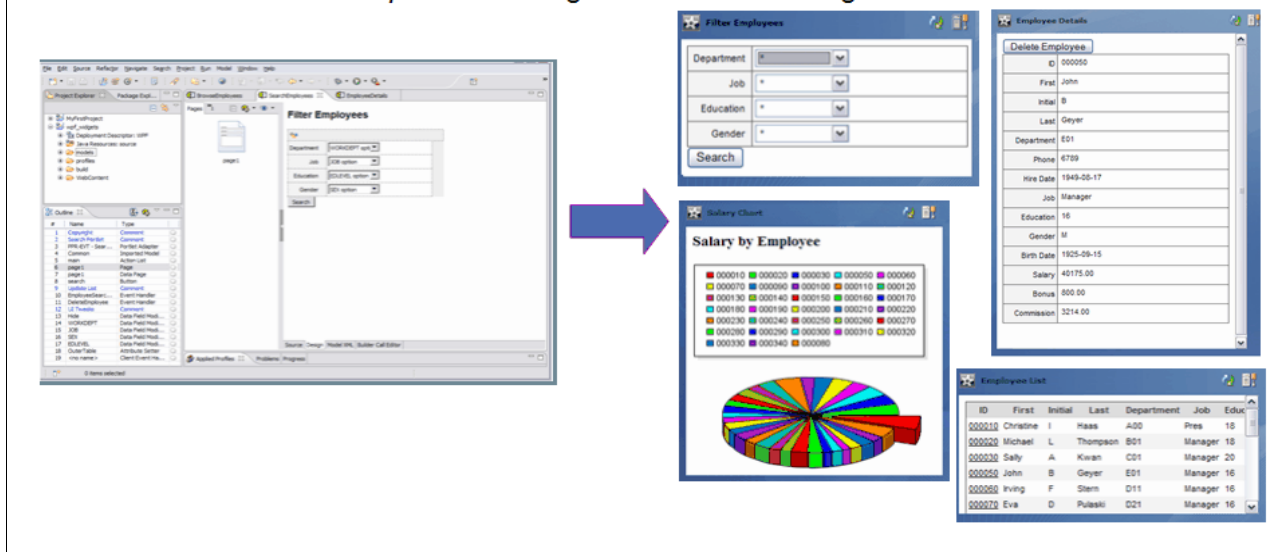


Figure 13-12 Lotus Mashup overview

13.3.3 WebSphere sMash

WebSphere sMash introduces a simple environment for creating, assembling and running applications based on popular Web technologies. Although not specially included in the IBM Mashup Center, it is considered part of the portfolio. It is based on the following Web technologies:

- ▶ A dynamic scripting runtime for Groovy and PHP. These are dynamic scripting languages
- ▶ Application programming interfaces optimized for producing REST services
- ▶ Rich Ajax Web user interfaces
- ▶ Integration mash-ups and feeds

WebSphere sMash has an associated incubation project, Project Zero, that enables the community to see the plans and direction of WebSphere sMash as it is being developed, and provide feedback and comments to direct the development effort.

13.4 IMS Web 2.0 Solutions architecture

During the development phase (see Figure 13-13 on page 284) of creating an IMS RESTful service and feed using the IBM Mashup Hub, the service developer must supply RDz-generated artifacts that are created by importing IMS Cobol copybook or a PLI source file. RDz-generated artifacts include the XML converter driver and the correlator file. The service developer then deploys the XML converters (generated into one driver file) to IMS Connect and creates an IMS feed in MashupHub by specifying IMS connection information and uploading the generated correlator file. The generated feed is automatically registered in

the MashupHub catalog. In addition, the field parameters extracted from the RDz-generated artifacts are registered as input parameters of the IMS RESTful service.

Developing an IMS Feed

- Develop an IMS RESTful service / feed using:
 - IMS Web 2.0 launchpoint in Rational Developer for System z
 - IMS Web 2.0 Editor in InfoSphere MashupHub

▪ SETUP steps:

- **RDz**
 - Import IMS COBOL or PL/I sources
 - Generate
 - XML converter driver
 - Correlator file
- **IMS Connect**
 - Install XML converter driver
- **IMS Web 2.0 in InfoSphere MashupHub**
 - Install correlator file
 - Create and set IMS feed connection information
 - Host Name or IP Address of the IMS host
 - Port number for IMS Connect
 - Data Store Name of the target IMS
 - SSL or user ID, password, group name
 - Add feed to MashupHub Catalog

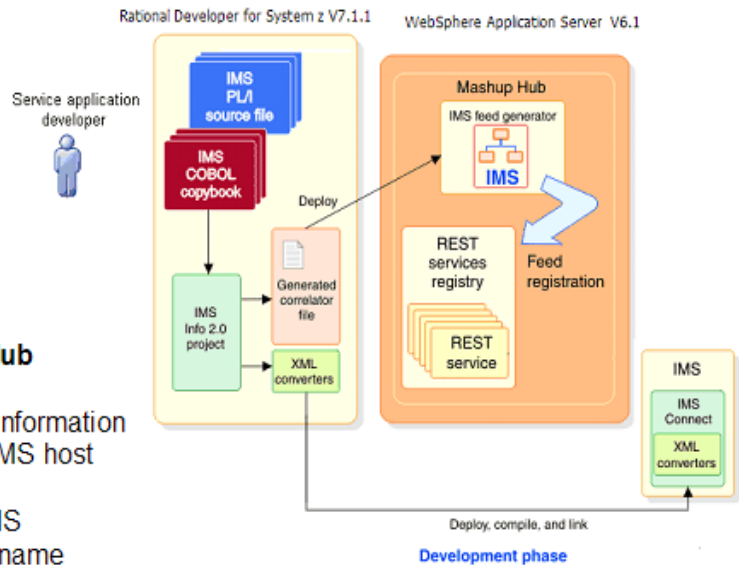


Figure 13-13 Developing an IMS Feed

During the runtime phase (see Figure 13-14 on page 285), the Web server receives an HTTP GET request from a Web client, and passes the request to the IBM Mashup Hub. The Mashup Hub finds the RESTful service registry and invokes the IMS RESTful service adapter, which processes the parameters, establishes a connection with IMS Connect, and generates a XML request to send to IMS Connect based on the name-value pairs.

The XML adapter function in IMS Connect converts the XML request message into byte arrays by using the XML converter driver (deployed by the service developer during development phase), and sends the data as an input message for the IMS transaction. The output byte array that is returned by the IMS transaction is converted by IMS Connect into XML response data by using the converter driver. The response is then returned to the IMS Web 2.0 Solution service adapter, which converts the response to the Atom format and sends it to the client as an HTTP response.

Running an IMS Feed

- Invoking an IMS RESTful service / feed
 - IMS Web 2.0 Generator in InfoSphere MashupHub
 - Mashup editing tools in InfoSphere MashupHub and Lotus Mashups

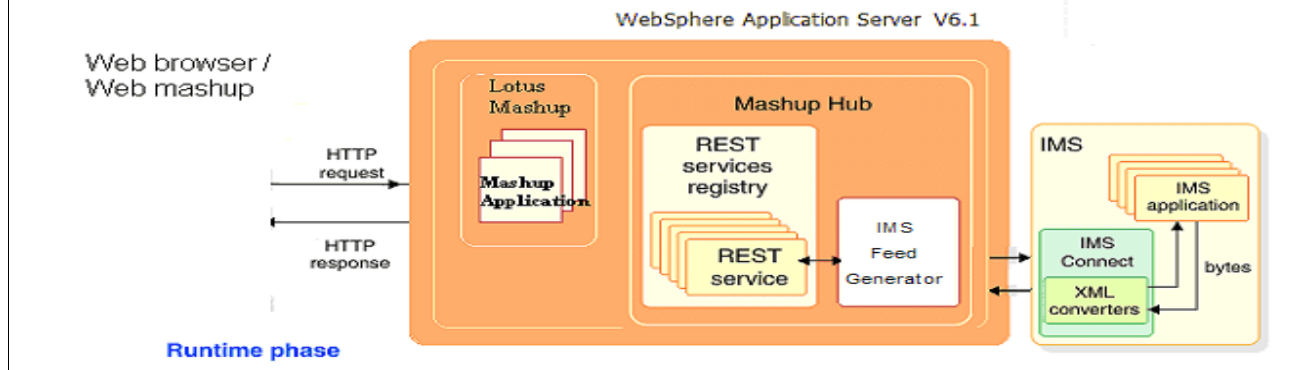


Figure 13-14 Running an IMS Feed

13.4.1 Software requirements and restrictions

As in every technological implementation there are base levels of the supporting software and certain restrictions that must be understood.

Software requirements

- ▶ IBM Mashup Center Enterprise Edition Version 1.0
- ▶ IBM IMS Version 10 integrated with IMS Connect Version 10 or later
- ▶ IBM Rational Developer for System z Version 7.1.1 or later
- ▶ Windows 2003 Server
- ▶ Internet Explorer V6, Internet Explorer V7, Firefox V2, and Safari V3

Restrictions

Each IMS application must have its own feed, and only non-conversational IMS applications that use the commit mode 1 (CM1), sync level NONE protocol, are supported.

IMS Web 2.0 Solutions currently has the following restrictions:

- ▶ Conversational applications are not supported.
- ▶ Two-phase commit is not supported.
- ▶ MFS source file is not supported. Only COBOL copybook and PL/I source files are supported.
- ▶ Multi-segment messages are not supported.
- ▶ Callout requests from IMS applications are not supported.

13.5 Creation of IMS RESTful service from an IMS application

You can create an IMS RESTful service or a feed from an IMS application by generating the XML converter driver and a correlator file first. Then you deploy the XML converter driver to IMS Connect, and upload the correlator file to MashupHub.

To create an IMS feed, you must have:

- ▶ IBM InfoSphere MashupHub Enterprise Edition.
- ▶ Your IMS application must be running on IMS Version 10 with integrated IMS Connect Version 10.
- ▶ IBM Rational Developer for System z Version 7.1.1 or later. Rational Developer for System z lets you create an IMS Web 2.0 project and guides you through the steps to generate the correlator file that correlates the input and output data structure from the IMS application with the appropriate XML format that the feed server understands. The tool also generates the XML converter driver that you deploy to IMS Connect, so IMS Connect can convert the responses and requests between MashupHub and IMS.

13.5.1 Identifying the application design and requirements for the feed

Work with your IMS application developer, application architect, and others team members to obtain and identify the necessary information about the requirements of the feed. Document key information that your users (including feed mashup creators) need to know; for example:

- ▶ What are the input and output flows for the IMS application?
Suppose that you have a phone book application. Do you want users of the feed to look up information only? Do you want them to be able to update, add, or delete any entry? Should the users be able to see the phone number and the address?
- ▶ Which input parameters are required to invoke the feed?
Does your application require that the value of a specific parameter be specified? Suppose that in a phone book application, users must specify a last name and a command of either DISPLAY, DELETE, ADD, or UPDATE to invoke the application. Without a last name and a command, the application does not run. In this case, both the last name and the command parameters are required.
- ▶ What should users be allowed to do with the feed? Which input parameters should be hidden from users so they cannot specify a value for the parameter when they invoke the feed?
Are there cases where you want to predefine a value, regardless of what the users specify? For example, in the phone book application, to limit your users to lookup requests only, hide the command parameter from your users. Because the command parameter is required to invoke the feed, assign it a default value of DISPLAY.

13.5.2 Generating Metafiles (XML converter driver and correlator file)

For messages to flow between MashupHub and IMS, an adapter is required to handle the translation of request and response messages. This adapter correlates the messages with the language structure of the IMS application, and invokes the XML converter driver on IMS Connect to convert the messages between XML and bytes.

The IMS Web 2.0 Solution service adapter is a plug-in in MashupHub that provides the necessary service between MashupHub and IMS. The IMS Web 2.0 Solution service adapter must have the following information:

- ▶ The inbound and outbound message structure of the IMS application
- ▶ The XML converter driver name to invoke on IMS Connect
- ▶ The IMS transaction and connection information needed to communicate with IMS Connect

Two files must be created from the IMS application source file (COBOL copybook or PL/I source file) by using Rational Developer for System z Version 7.1.1 or later:

- ▶ The XML converter driver

The XML converter driver is generated based on the IMS application source file that describes the input and output language structure. Each input or output language structure needs its own converter. You must use Rational Developer for System z to generate all the converters for an IMS application into one driver file. The driver must be deployed to IMS Connect.

IMS Connect has a XML adapter function that converts the request and response messages between XML and bytes based on your specified XML converter driver.

- ▶ The correlator file

The correlator file contains information that is required for the IMS Web 2.0 Solution service adapter to communicate with IMS Connect.

Figure 13-15 lists the information to specify in Rational Developer for System z.

Information	Description
Your IMS application source file	The Enterprise Service Tools wizard parses the application source file to identify the language structures.
z/OS as the platform that the application runs on	The platform selection will impact other platform information attributes, such as floating point format and endian, to default values that are appropriate for the z/OS platform.
The inbound language structure in your application	Based on the language structures from your application, the Enterprise Service Tools wizard needs to know which data structure is the input to invoke the application.
The outbound language structure in your application	Based on the language structures from your application, the Enterprise Service Tools wizard needs to know which data structure is the output from the application.
The prefix for the converter name	The Enterprise Service Tools wizard generates the converters based on the prefix. Because the Enterprise Service Tools wizard lets users choose to generate separate converter files for inbound and outbound messages, it appends a different character to the prefix to create different converter files.
Character encodings	The code page to use for encoding the inbound and outbound XML messages, and the code page that is used by the z/OS host system.
Transaction code for your IMS application	The transaction code is required to run the IMS application.
Socket timeout and execution timeout values	The time the IMS Info 2.0 service adapter should wait for a response from IMS Connect, and for IMS Connect to send a message to IMS and receive a response before timeout.

Figure 13-15 Information to specify in Rational Developer for System z

Example 13-1 shows the generated correlator file for the phonebook sample application.

COBOL application programs: For COBOL application programs, the dash symbol (–) in field names is represented as an underscore (_) in the corresponding XML tags. For example, a field named IN-LL is represented in XML by the opening and closing tags <in_ll> and </in_ll>.

Example 13-1 Generated correlator file for the phone book sample application

```
<?xml version="1.0" encoding="UTF-8"?>
<IMS:IMSInfo20Correlator xmlns:IMS="http://www.ibm.com/IMS/IMSInfo20Correlator"
generator="com.ibm.etools.xmlent.common.xform.gen"
generatorVersion="7.1.200.V200712031646" version="1.0">
  <IMS:correlatorEntry>
    <IMS:converterName>IMSPHBD</IMS:converterName>
    <IMS:socketTimeout>6000</IMS:socketTimeout>
    <IMS:executionTimeout>5000</IMS:executionTimeout>
    <IMS:ltermName></IMS:ltermName>
    <IMS:trancode>IVTNO</IMS:trancode>
    <IMS:inboundTPIPENAME></IMS:inboundTPIPENAME>
    <IMS:inboundCCSID>1208</IMS:inboundCCSID>
    <IMS:hostCCSID>1140</IMS:hostCCSID>
  </IMS:correlatorEntry>
</IMS:IMSInfo20Correlator>
```

```

    <IMS:outboundCCSID>1208</IMS:outboundCCSID>
  </IMS:correlatorEntry>
  <IMS:securityProperties>
    <IMS:keyStoreName></IMS:keyStoreName>
    <IMS:keyStorePasswd></IMS:keyStorePasswd>
    <IMS:trustStoreName></IMS:trustStoreName>
    <IMS:trustStorePasswd></IMS:trustStorePasswd>
  </IMS:securityProperties>
  <IMS:serviceParameters>
    <IMS:parameter index="1" maxchars="4" name="in_ll" value=""
      xpath="/INPUTMSG/in_ll[1]" />
    <IMS:parameter index="2" maxchars="4" name="in_zz" value=""
      xpath="/INPUTMSG/in_zz[1]" />
    <IMS:parameter index="3" maxchars="10" name="in_trcd" value=""
      xpath="/INPUTMSG/in_trcd[1]" />
    <IMS:parameter index="4" maxchars="8" name="in_cmd" value=""
      xpath="/INPUTMSG/in_cmd[1]" />
    <IMS:parameter index="5" maxchars="10" name="in_name1" value=""
      xpath="/INPUTMSG/in_name1[1]" />
    <IMS:parameter index="6" maxchars="10" name="in_name2" value=""
      xpath="/INPUTMSG/in_name2[1]" />
    <IMS:parameter index="7" maxchars="10" name="in_extn" value=""
      xpath="/INPUTMSG/in_extn[1]" />
    <IMS:parameter index="8" maxchars="7" name="in_zip" value=""
      xpath="/INPUTMSG/in_zip[1]" />
  </IMS:serviceParameters>
  <IMS:nameValuePairsToXmlTemplate><![CDATA[
<INPUTMSG>
  <in_ll>&in_ll;</in_ll>
  <in_zz>&in_zz;</in_zz>
  <in_trcd>&in_trcd;</in_trcd>
  <in_cmd>&in_cmd;</in_cmd>
  <in_name1>&in_name1;</in_name1>
  <in_name2>&in_name2;</in_name2>
  <in_extn>&in_extn;</in_extn>
  <in_zip>&in_zip;</in_zip>
</INPUTMSG>
]]&gt;</IMS:nameValuePairsToXmlTemplate>
</IMS:IMSInfo20Correlator>

```

13.5.3 Deploying the XML converter driver to IMS Connect

Figure 13-16 presents an overview of the development phase of the IMS Web 2.0 Solution.

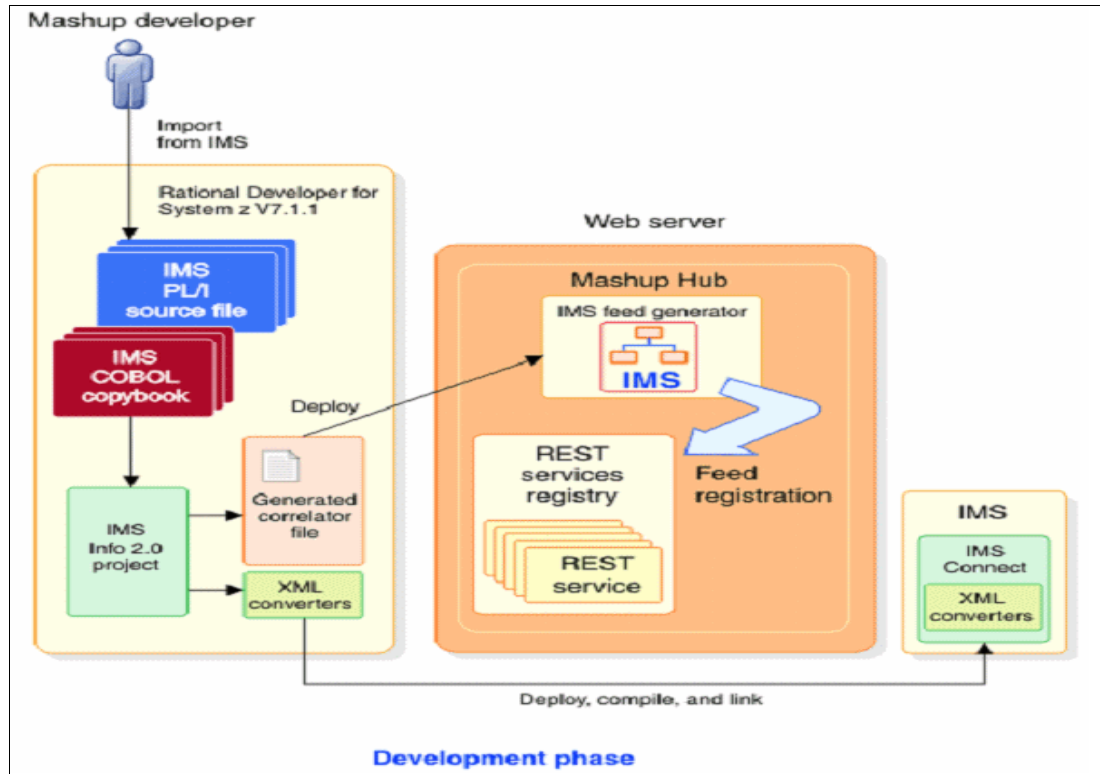


Figure 13-16 The development phase of the IMS Web 2.0 Solution

The steps to deploy the XML converter driver include:

- ▶ FTP the XML converter driver that is generated by Rational Developer for System z to your host data set.
- ▶ Configure the IMS Connect configuration member:
 - Add the ADAPTER configuration statement, ADAPTER=(XML=Y), in the IMS Connect configuration member.
 - Specify the HWSSOAP1 user message exit in the EXIT= parameter of the TCP/IP configuration statement.
 - Define the XML adapter as a Base Primitive Environment (BPE) exit routine for IMS Connect by coding a BPE exit list PROCLIB member. Then point to it through the EXITMBR statement in the BPE configuration PROCLIB member.

z/OS Unicode Conversion Services must be configured to support character conversions from UTF-8 to EBCDIC and from EBCDIC to UTF-8. Contact your z/OS system administrator for this task.

- ▶ Compile and link the XML converter driver into a data set that is concatenated to the STEPLIB in the IMS Connect startup JCL. You are ready to create an IMS feed in MashupHub.

When you link the XML converter driver, set the program name to IMSPHBKX. The program name is an ALIAS in the link job for the Converter Metadata Aggregate Service. The Converter Metadata Aggregate Service program name has the same name as the converter, except that the last character is an X.

13.5.4 Creating a feed from an IMS application in InfoSphere MashupHub

The major steps in this implementation phase are:

1. Specify the IMS host information (hostname, data store, and port number) for the data source.
2. Upload the correlator file that you created by using Rational Developer for System z.
3. Provide the security information (RACF and/or SSL) to create an IMS feed.
4. Select to hide the parameters that you do not want feed consumers to be able to set a value for when they invoke the feed.
5. Specify the required information to identify this feed (title, description, version).
6. Test and invoke your IMS feed.

There are two ways to invoke an IMS feed:

- ▶ Use the View Feed link in the Actions pane in MashupHub.
- ▶ Specify the URL of the feed directly in a browser.

If your feed accepts input parameters, the values for the input parameters can be specified in the URL as name-value pairs in the following format:

```
http://url_to_your_feed?parameter_list
```

The parameter_list consists of name-value pairs separated by an ampersand (&), for example:

```
parm1=value1[&parm2=value2&parm3=value3...]
```

Depending on your browser, what you see as a result might be different:

- ▶ In Internet Explorer, the output XML containing actual data from the feed source is displayed.
- ▶ In Mozilla Firefox, instead of the XML, the title and other meta information of the feed are displayed. To see the output XML with actual data, in Mozilla Firefox right-click and select View Page Source.
- ▶ In Safari, instead of the output XML, the title and other meta information of the feed are displayed.

Sample format for IMS feeds

The output for an IMS feed is in the Atom feed format, and the output of the original IMS transaction is placed within a single repeating element entry, rather than multiple repeating elements within the Atom feed.

Because the output from an IMS feed is highly dependent upon the original IMS transaction, and the output can be a mix of array entries and single-field elements, the output XML does not translate to the multiple repeating element format of an Atom feed. The output XML contains:

- ▶ Only one occurrence of the <RepeatingElement> element.
- ▶ The repeating element contains two singular fields that are in every IMS feed output, the LL and ZZ fields.
- ▶ The repeating element also includes the data of interest, which can a mix of array entries and single-field elements.

“Feed from an IMS application in MashupHub” on page 362 displays a sample output from an application that returns information on multiple insurance policy holders.

After an IMS feed is created, because IMS feeds return output in arrays, to display the data properly in a mashup, you need to use the transform operator in the MashupHub feed mashup builder to restructure the arrays into individual entries.

For more information on the IBM Mashup Center, refer to the Web site:

<http://www-01.ibm.com/software/info/mashup-center/>

13.6 IMS Web 2.0 and IMS On-demand

IMS Web 2.0 Solution is yet another way of accessing IMS, in addition to using IMS TM Resource Adapter and IMS SOAP Gateway, as shown in Figure 13-17. It leverages existing IMS SOA infrastructures to provide a lightweight services access to IMS transactions.

With the help of Lotus Mashup and its ability to create Web mashups, IMS Web 2.0 Solution provides users with a faster way of developing B2C solutions.

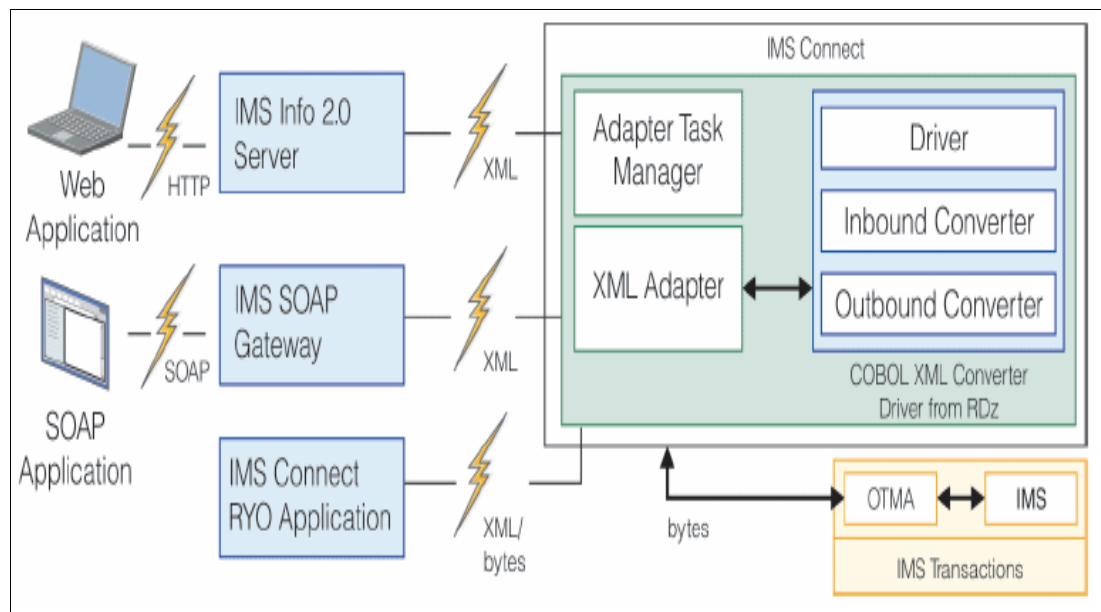


Figure 13-17 The IMS Web 2.0 Solution option to access IMS

This concludes our discussion about the IMS Web 2.0 Solution



DLIModel Utility, DB Web Services, and XQuery

In this chapter, we discuss the new tooling support in the DLIModel utility, which enables IMS Version 10 Clients to easily define and generate deployable Web services capable of simple IMS database operations, including data retrieval, inserts, updates, and deletes. The metadata produced by the DLIModel utility now also supports XML exploitation of the hierarchical organized DLI data.

IMS data, with no type enforcing catalog and the inherent differences of encoding and byte ordering on the zSeries®, is not easily shared across the industry or enterprise. This limits IMS' ability to share data across heterogeneous environments. There is a need to easily share IMS data despite language and platform dependencies. XML became the de facto data interchange language to resolve this situation with IMS.

We show how to configure Java applications for accessing IMS databases both locally or remotely.

We also describe the new Web Services for DLI databases.

14.1 IMS databases in the SOA environment

Here are some important characteristics of access to IMS databases in the SOA environment:

- ▶ The following can access DLI databases through the use of two API's (assisted DLI access and JDBC):
- ▶ Java programs running in IMS Java Message Program (JMP) or Java Batch Program (JBP) dependent regions
- ▶ Java programs in the WebSphere Application Server
- ▶ Stored procedures in DB2
- ▶ Programs in a CICS region

This requires the assistance of IMSDLI java classes (JDR adapter) for JMP and JBP regions, or the use of the DRA and DB resource adapter for DLI, for the others.

The DRA and DB resource adapter allows external partners on the same z/OS image to directly access the database component of an IMS subsystem. These interfaces are shown in Figure 14-1.

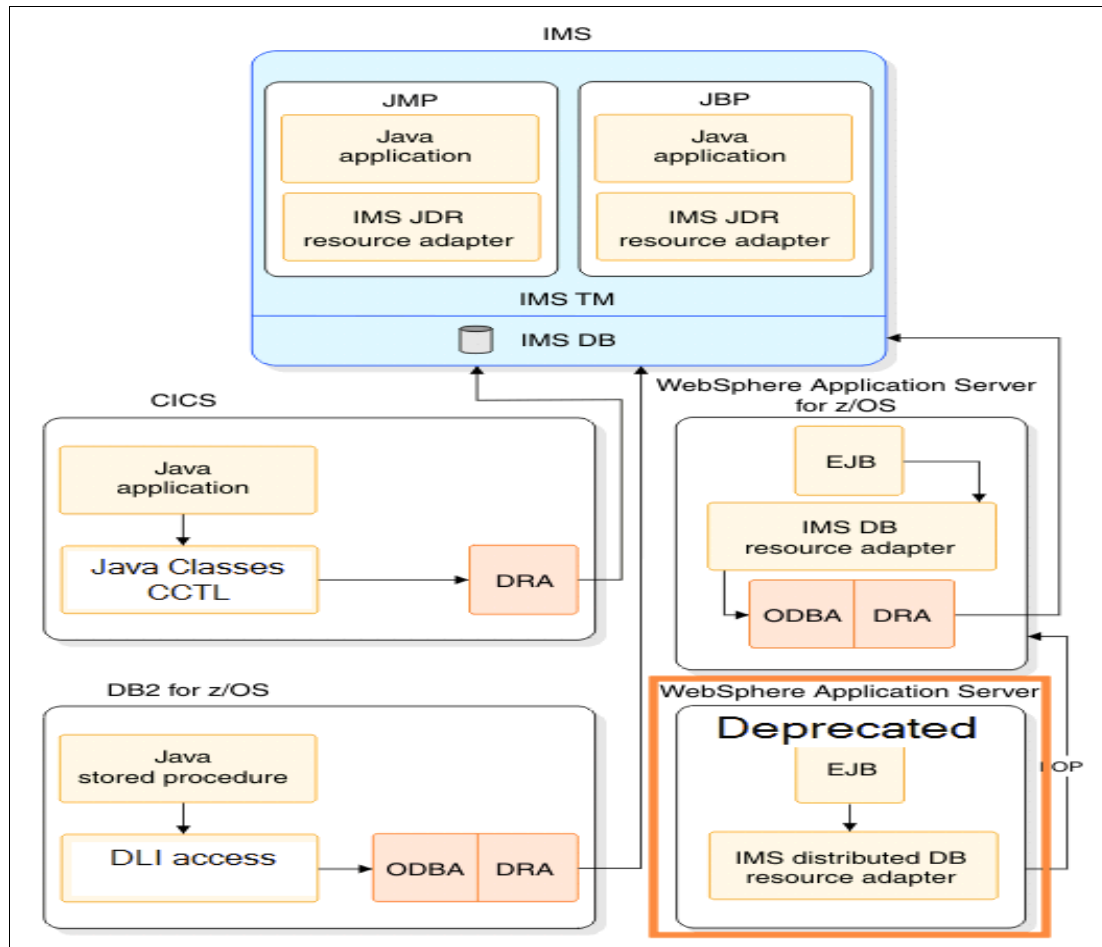


Figure 14-1 IMS JDR and DRA adapter support

In Figure 14-1 the “IMS remote DB resource adapter”, uses the local DRA adapter through a session Enterprise Java Bean (EJB) connection. The DRA adapter and EJB are deployed in a WebSphere Application Server. This solution is considered as deprecated because it is not JCA compliant.

In IMS SOA solutions there is a requirement for metadata describing the DLI databases, and relationships to secondary indexes, and logically related databases. Providing this metadata is the role of the DLIModel utility. For both API's this metadata is required. The DLIModel utility provides a DLIDatabaseView metadata class. Figure 14-2 presents reasons why the DLIModel utility is required.

- **IMS does not have a runtime metadata catalog**
 - Java libraries for IMS require meta-information to properly abstract IMS database information
 - JDBC (SQL)
 - XML-DB (SQL UDFs)
 - XQuery (FLWOR expressions)
- **Needed a way to represent IMS database metadata for consumption at runtime**
 - PSB information
 - PCBs
 - DBD information
 - Hierarchy, segments, key/search fields
 - Copybook information
 - Additional fields, field types, etc

Figure 14-2 Reasons for utilizing the DLIModel utility

DLI databases are hierarchical and are extremely suitable to store and to build XML documents. In DLI, a database segment definition defines the fields for a set of segment instances similar to the way a relational table defines columns for a set of rows in a table. In this way segments relate to relational tables, and fields in a segment relate to columns in a relational table. The name of an IMS segment becomes the table name, and the name of a field becomes the column name in the SQL query.

A fundamental difference between segments in a hierarchical database and tables in a relational database is that, in a hierarchical database segments are implicitly joined with each other. In a relational database you explicitly join two tables. Looking at the hierarchical data, we can consider that all dependent segments have a foreign key, which is a concatenation of the keys its parent path. This is shown in Figure 14-3 on page 296.

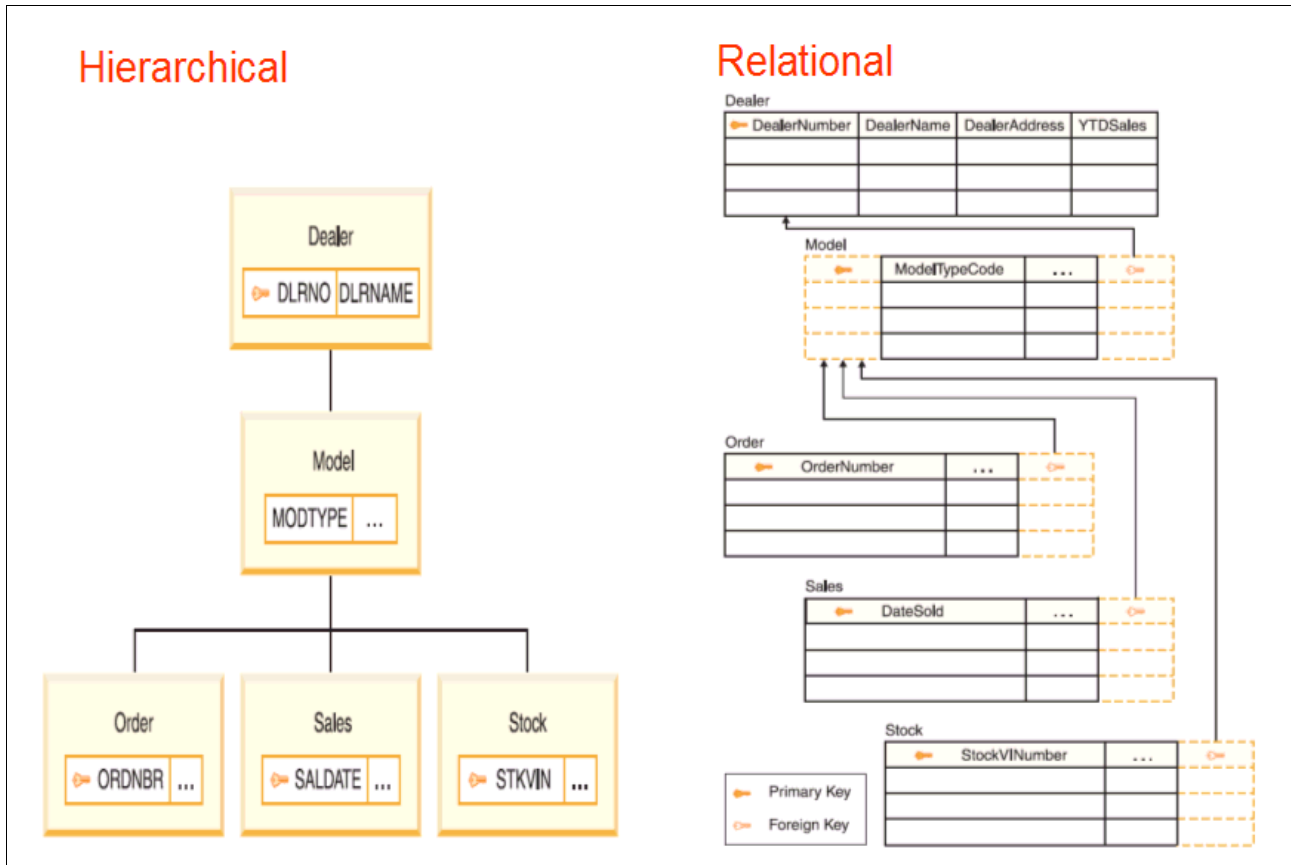


Figure 14-3 Implicit foreign KEYS with DLI

A segment instance in a hierarchical database is already joined with its parent segment and its child segments along the same hierarchical path. In a relational database this relationship between tables is captured by foreign and primary keys. To be used with JDBC, this hierarchical view must be transformed logically in a relational view. This happens by concatenating all segments in its hierarchical path. The key of a segment/row in the relational view is a concatenation of all keys in its path.

The solutions that IMS for Java development offer are:

- ▶ The IMS Java dependent region (JDR) adapter

This adapter is installed using SMP/E with IMS (FMID JMK1016 in IMS Version 10), and is the set of Java class libraries that allow you to write Java application programs that run under IMS on the host in JBP and JMP regions. These libraries offer traditional IMS application program support and allow Java application programs to access the same DL/I functionality provided in MPP and non-message driven BMP regions, such as:

- Accessing IMS databases
- Accessing IMS message queues to read and write messages
- Performing program switches
- Commit and rollback processing

The Java API for IMS DB offers more granular database access than the JDBC driver. For example, it allows IMS developers to build SSAs and issue any of the database DL/I calls directly.

- ▶ Solutions, using the data resource adapter (DRA) to access IMS DLI, composed of Java Classes bundled with the IMS DB resource adapter. These are a set of Java class libraries that allows you to write Java application programs that access IMS databases from:
 - WebSphere Application Server for z/OS
 - DB2 for z/OS stored procedures
 - CICS Transaction Server for z/OS
 - IMS on the host in JBP and JMP regions (this support is also included in the IMS Java dependent region resource adapter)

As shown in Figure 14-1 on page 294 some solutions use the ODBA API. The IMS DB resource adapters include a type-2 JDBC driver for all of the above environments, which implements the JDBC 2.0 API. The JDBC driver offers SQL support for IMS databases.

For IMS database support, the libraries offer complete assisted DL/I support, segment search argument support, command code support, database read/write support for all database types, and datatype conversion.

- ▶ Besides the above mentioned adapters, a new IMS distributed DB resource adapter, which replaces the deprecated remote solution, are made available. Details are explained in Chapter 15, “IMS Open Database and Universal Drivers” on page 321.

But before using the API's, if you want to program DLI access in Java, it is required to build the metadata information for the DLI databases. This is performed with the DLIModel utility.

Information about Java Programming can be found in:

- ▶ *IMS Java Guide and Reference Version 9*, SC18-7821
- ▶ *IMS Application Programming Guide Version 10*, SC18-96988

14.2 Configuring for Java applications accessing IMS DB

To access DLI data from the WebSphere Application Server we have three solutions:

- ▶ From WebSphere Application Server on z/OS
- ▶ From WebSphere Application Server on distributed over TCP/IP to IMS Connect and connection to Open Data Base Manager (ODBM)
- ▶ From WebSphere Application Server on distributed with connections to EJBs on a local WebSphere Application Server on z/OS (older and deprecated)

14.2.1 DB Resource Adapters (IMS Version 11 and IMS Version 10 SPE)

There are times when your services or other non-IMS applications require access to IMS data without executing an IMS transaction. The IMS DB Resource Adapter provides this function. This Resource Adapter can be used in a number of environments: it can be installed on either the z/OS or distributed WebSphere Application Server.

There are three versions of the DB Resource Adapter:

- ▶ z/OS Database Resource Adapter.
- ▶ Distributed Resource Adapter for IMS Connect/ODBM.
- ▶ Distributed Resource Adapter, available already in IMS Version 9, which require a remote and local WebSphere Application Server. This solution is deprecated, and we do not further discuss it again in this publication.

IMS provides two versions of the new IMS Universal DB resource adapters for optimized transaction management and performance.

- IMS Universal DB resource adapter with local transaction support

This resource adapter provides local transaction support when deployed on any supported Java EE application server.

- IMS Universal DB resource adapter with XA transaction support

This resource adapter provides both XA transaction and local transaction support when deployed on any supported Java EE application server connecting to IMS databases.

Use the IMS Universal DB resource adapter with XA transaction support for global or two-phase commit functionality. For one-phase commit functionality, the IMS Universal DB resource adapter with XA transaction support can be used, however performance might be improved by using the IMS Universal DB resource adapter with only local transaction support.

To provide for different transactional qualities of service for Java EE applications, it is possible to deploy both resource adapters into the same Java EE application server. When multiple resource adapters are used in the same Java EE application server, they must all belong to the same version of IMS.

14.2.2 Accessing DLI data on z/OS

There is more than one path to access DLI data bases in the SOA environment:

- ▶ Access from WebSphere Application Server
- ▶ Access through ODBM on request from a distributed client

Access from WebSphere Application Server

To access DLI data from a WebSphere Application Server for z/OS, you must install the IMS DB Resource Adapter (the Java class libraries for IMS) on WebSphere Application Server for z/OS, and configure both IMS open database access (ODBA) and the data resource adapter (DRA). This is also true when the WASs on z/OS is used as the local accessor for a distributed WebSphere Application Server.

The JDBC or IMS hierarchical database interface for Java calls are passed to the IMS DB resource adapter, which converts the calls to DL/I calls. The IMS DB resource adapter passes these calls to ODBA, which uses the DRA to access the DL/I region in IMS.

To use the IMS DB resource adapter with WebSphere Application Server for z/OS, you must use WebSphere Application Server Version 6.0 for z/OS or later. You must also use RRS (resource recovery services) for z/OS. The IMS DB resource adapter supports the 64-bit addressing mode of WebSphere Application Server for z/OS version 6.1.04

The ODBA interface uses the Database Resource Adapter (DRA) to communicate with IMS DB. To set up ODBA and the DRA, follow these steps:

1. Create the ODBA DRA startup table. The create of the ODBA DRA startup table is done by using the DFSPRP macro. This is explained in following document:

- *IMS V10 Communications and Connections Guide*, SC18-9703

2. Set up resource access security (RAS)

A security check is performed by RACF to determine if the user is authorized to use the PSB. RACF determines authorization by looking at the RACF security class profile defined for the dependent region.

The Database Resource Adapter must be installed from the WebSphere Application Server for z/OS administrative console from a RAR file, which is located in:

pathprefix/usr/lpp/ims/imsjava10/imsDBJCA.rar

In the RAR file we find the following jar files:

```
pathprefix/usr/lpp/ims/imsjava10/imsJDBC.jar
pathprefix/usr/lpp/ims/imsjava10/imsDBJCA.jar
pathprefix/usr/lpp/ims/imsjava10/imsjavabase.jar
pathprefix/usr/lpp/ims/imsjava10/imsXQuery.jar
```

The `imsXQuery.jar` file is required only if you intend to use XQuery.

The binding elements between your program and the databases are:

- ▶ The `DLIDatabaseView` metadata class, produced by the `DLIModel` utility.
- ▶ The name of the Data Resource Adapter (DRA) which makes the connection to IMS over ODBA.

Note: Application programs, running under WebSphere Application Server can use a managed Connection Factory. The connection to IMS over DRA was described with the Admin Console under the Data Base Resource Adapter, and can be located through the JNDIname.

Access from ODBM (IMS Version 11 and IMS Version 10 through a SPE)

Database connection and access requests are routed to ODBM from the distributed resource adapters and APIs through IMS Connect. ODBM routes the database connection and access requests received from IMS Connect to the IMS systems that are managing the requested database. Figure 14-4 on page 300 presents this flow.

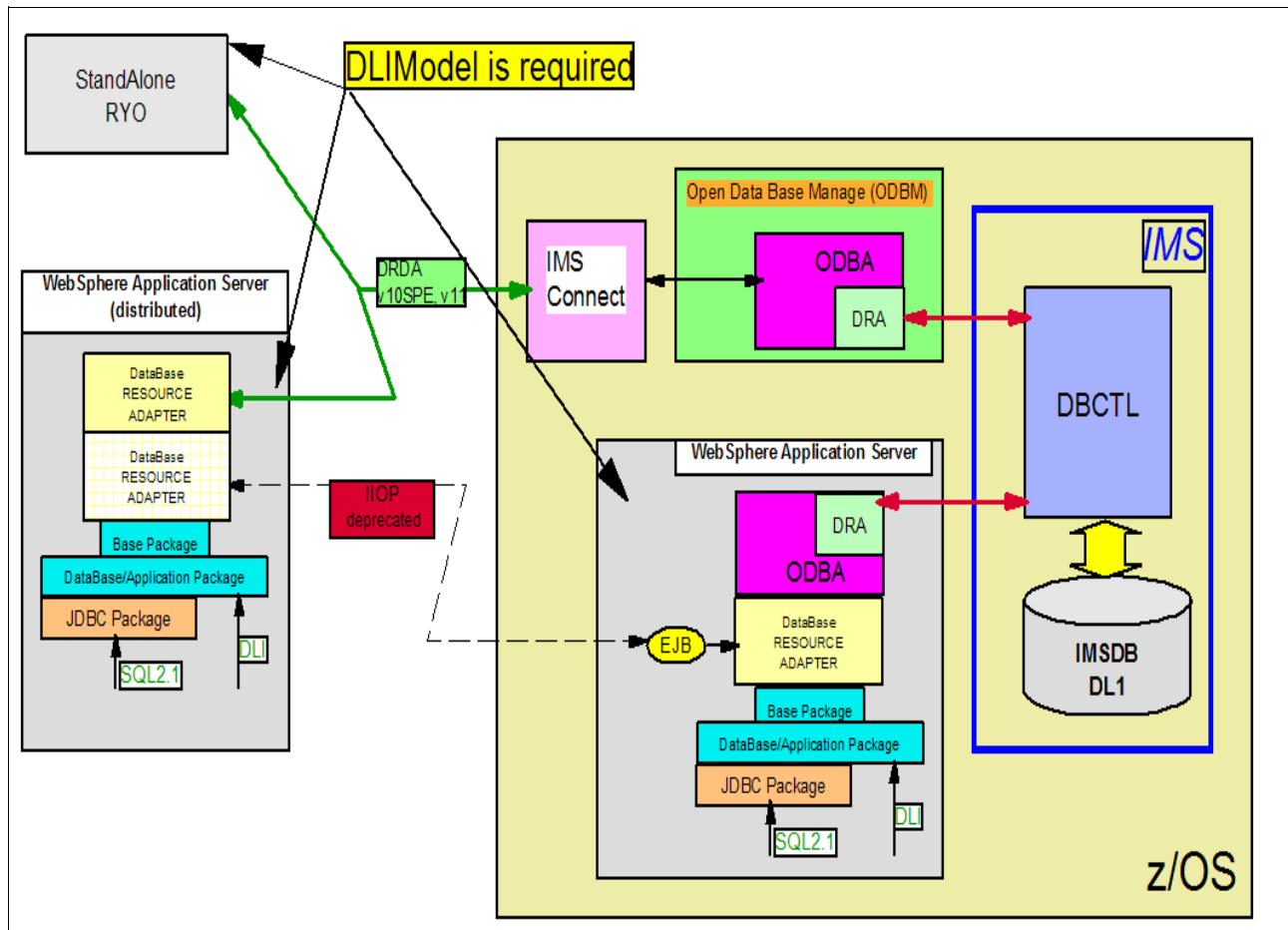


Figure 14-4 Distributed DLI access thru IMS Connect and ODBM

ODBM translates the incoming database requests from the Distributed Data Management (DDM) architecture that is a part of DRDA into the DL/I calls expected by IMS. When ODBM returns the IMS output to the client, ODBM translates the response back into the DDM protocol.

14.2.3 Accessing IMS data from distributed environments

To access IMS data from a WebSphere Application Server for distributed, you must install the IMS DB Resource Adapter (the Java class libraries for IMS) on this WebSphere Application Server. Use the IMS Universal JDBC driver in type 4 driver mode for Java programs that access the IMS subsystem through a TCP/IP network connection.

The distributed DB resource adapter comes as a RAR file with IMS and must be transferred to the platform, where it must be installed. The IMS Universal JDBC driver is an IMS Connect client.

The binding elements between your program and the databases are:

- ▶ DLIDatabaseView class, which is produced by the DLIModel utility represents the detailed meta information about one PSB with PCB's, and accessed DBDs.
- ▶ Alias name of the IMS known by ODBM, which is mapped to the ODBA access into a particular IMS.

- ▶ IP Address and port number of an IMS Connect DRDA port

Note: Standalone application programs written in Java can also access the DLI data, but must use nonManaged Connections because all connect information is described at runtime.

14.3 DLIModel Utility

In order for a Java application to access an IMS database, it requires information about that database. The DLIModel utility allows you to transform your IMS database information (program specification blocks, database descriptions, and COBOL copybooks) into application independent metadata.

The DLIModel utility parses PSBs, DBDs, and COBOL copybook information, and constructs an internal model of IMS metadata. The utility performs on one PSB at the time.

The constructed metadata model serves two purposes:

- ▶ To construct a visual representation of IMS resources for programming and DBA references.
- ▶ To construct required metadata for Java or IMS XML database application development.

Figure 14-5 presents the features available from the DLIModel utility since IMS Version 8.

Evolution of the DLIModel Utility

- **IMS Version 8**
 - z/OS batch utility and command line via UNIX System Services tool
 - The utility requires control statements but generates Java metadata classes
 - Also generates text based visualization
- **IMS Version 9**
 - Additional control statements are added for XML DB support
 - XML schema generated
- **IMS Version 10**
 - GSAM database support added
 - Enhanced XML schema generation for XQuery support
- **Currently**
 - Eclipse 3.x based GUI
 - Visualizes an entire IMS PSB
 - PSB and DBD XMI based metadata generation
 - Control statements can be imported from previous generations
 - Allows direct COBOL copybook import with RAD or RDz
 - Generates EAR and WSDL files for IMS V10 DB Web Services

Figure 14-5 Evolution of the features available from the DLIModel utility

Documentation on the DLIModel utility can be located in *IMS System Utilities Reference Version 10*, SC18-9968.

In addition to creating metadata, the DLIModel utility also performs:

- ▶ Generates annotated XML schema for IMS databases, based on selected PCB definitions. These XML schema define the XML layout used when retrieving XML data from or storing XML data in IMS databases.
- ▶ Users can now import COBOL copybook directly into the DLIModel utility if they have RAD or RDz.
- ▶ Incorporates additional PCB, segment, and field information, or overrides existing information.
- ▶ Generates a database report, which is designed to assist application developers. The report is a text file that describes a particular PSB and all PCBs. This includes segment names, field names, field types, database hierarchies, and other database specific information.

Figure 14-6 illustrates the input and output elements associated with the DLIModel utility process.

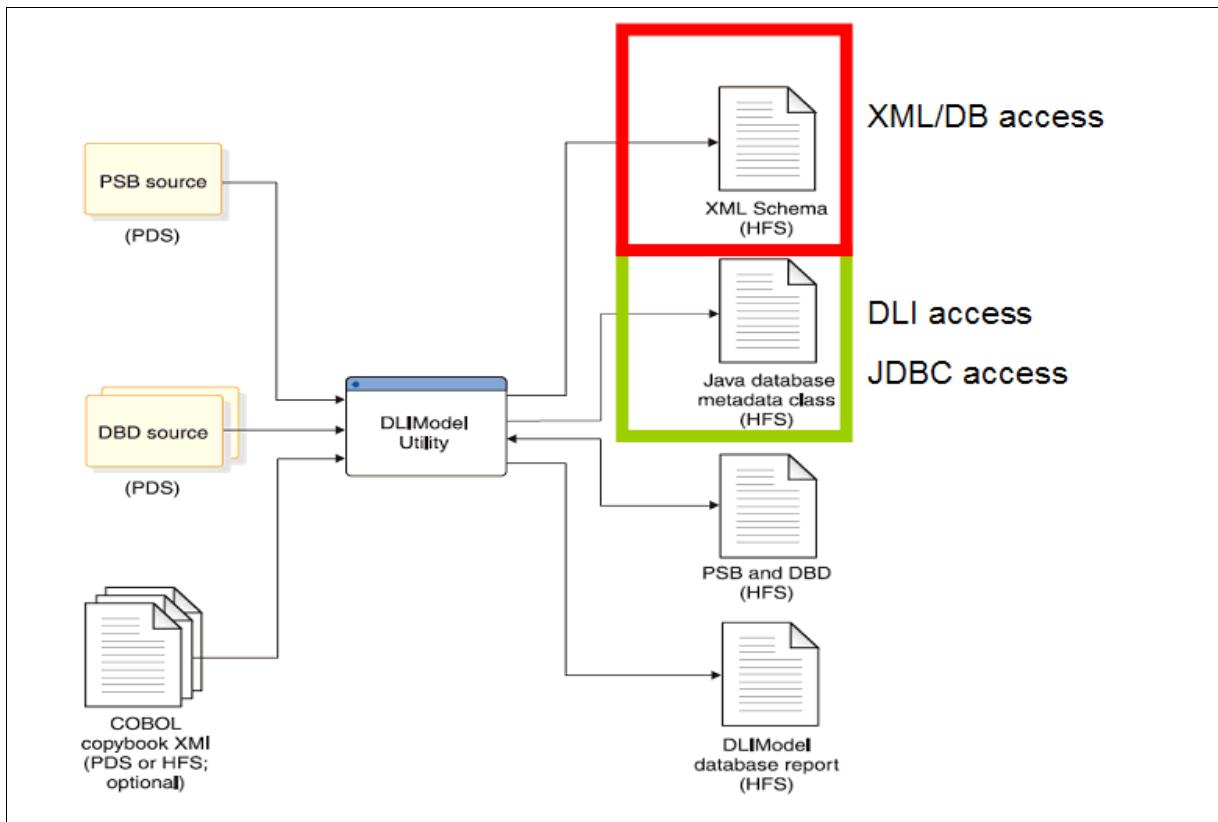


Figure 14-6 Input and Output of the DLIModel utility

The DLIModel utility can process:

- ▶ All database organizations except MSDB, HSAM, and SHSAM
- ▶ All types and implementations of logical relationships
- ▶ Secondary indexes, except for shared secondary indexes
- ▶ Secondary indexes that are processed as stand-alone databases

- ▶ PROCOPT=K option in a PSB SENSEG statement. If a segment has PROCOPT=K specified, an unqualified Get Next call (GN) skips to the next sensitive segment with a PROCOPT other than K.
- ▶ PSBs that specify field-level sensitivity

The DLIModel utility is a Java application, so you can run it from:

- ▶ The UNIX System Services prompt
- ▶ The z/OS-provided BPXBATCH utility

The Eclipse-based DLIModel utility plug-in obtainable from

<http://www.ibm.com/software/data/ims/toolkit/dlimodelutility>

Recommendation: IMS Version 10 is the last release to support the z/OS based batch DLIModel utility. The DLIModel utility plug-in requires Eclipse Version 3.2.2. Additionally, if the user has RAD or RDz, then the COBOL Import function is enabled.

The actions of the DLIModel utility are directed by control statements that you supply. Control statements are only used with the command line version. In the GUI, you can specify:

- ▶ Which PSB to process during a run
- ▶ Aliases for the PSB, PCBs, segments, and fields
- ▶ Data types and format masks for fields
- ▶ XMI files that contain XMI descriptions of COBOL copybook members for segments
- ▶ Additional field definitions for fields that are not defined in the DBD or the COBOL copybook files
- ▶ Information that overrides PSB, DBD, and COBOL copybook XMI information

If you have an existing IMS control statement to use with the version of the DLIModel utility that is shipped with IMS, you can reuse your control statement as source for generating metadata instead of individually specifying PSB and DBD files.

14.3.1 Using the DLIModel Utility plug-in

When you use the Eclipse plug-in, start by creating a “DLIModel utility Project” as displayed in Figure 14-7.



Figure 14-7 DLIModel utility Project

For this project specify:

- ▶ Name
- ▶ Package
- ▶ Advanced option (Control statement file or optional output)
- ▶ 1 Input PSB and DBDs of accessed DBs by PCBs in PSB

As displayed in Figure 14-7 one of the outputs produced by the DLIModel utility is a *DLIDatabaseView java class*. When using the IMS Open Database APIs, or the JDBC driver for IMS, or the DLI assisted API you need to provide the application view information of the databases to access a set of IMS databases. The application view information is in the program specification block (PSB), but you must convert it into a form that you can use in your Java application, which is a subclass of *com.ibm.ims.db.DLIDatabaseView*.

The subclass of *com.ibm.ims.db.DLIDatabaseView* is called the Java database metadata class. The DLIModel utility generates the metadata class by using the application PSB source and related DBD and other source files.

When you establish the database connection, you pass the name of this subclass to the resource adapter or JDBC driver. The Java database metadata class provides a view of a PSB and its related Program Control Blocks (PCBs) that the Java class libraries use at runtime to process both SQL and Java-based DLI calls.

When the following code is executed, DLIDriver; a class in com.ibm.ims.db, is loaded

```
Class.forName("com.ibm.ims.db.DLIDriver");
```

When the following code is executed, the JDBC Driver Manager object determines which of the registered drivers supports the supplied string:

```
connection = DriverManager.getConnection
("jdbc:dli:dealership.application.DealerDatabaseView");
```

A report is also generated, an excerpt of which is shown in Example 14-1. Note that you have a section for each PCB in the PSB. That section shows the exact hierarchy of "SENSE" segments, while it indicates the fields, key fields, secondary indexes.

Example 14-1 DLIModel report

```
DLIModel IMS Java Report
=====
Class: AUTPSB11DatabaseView in package: dealership generated for PSB: AUTPSB11
=====
PCB: AUTOLPCB
=====
Segment: DEALER
  Field: DLRNO      Type=CHARLength=4++ Primary Key Field ++
  Field: DLRNAME   Type=CHARLength=30      (Search Field)
  Field: CITY      Type=CHARLength=10      (Search Field)
  Field: ZIP       Type=CHARLength=10      (Search Field)
  Field: PHONE     Type=CHARLength=7       (Search Field)
=====
  Segment: MODEL
  .....
    Segment: ORDER1
    Segment: SALES
    Segment: STOCK
      Field: STKVIN Type=CHAR Length=20++ Primary Key Field ++
    Segment: STOCSALE
    Segment: SALESPER
    Segment: SALESINF
    Segment: EMPLINFO
  .....
=====
PCB: AUTS1PCB
```

```

=====
Segment: ORDER1
  Field: ORDNBR    Type=CHARLength=6    (Search Field)
.....
Field: XFLD1++ Secondary Key Field ++
  Component search fields:
  LASTNMType=CHARLength=25
  FIRSTNMType=CHARLength=25
  ORDNBRTType=CHARLength=6
  Segment: MODEL
    Segment: DEALER
    Segment: STOCK
=====
PCB: AUTS2PCB
=====
Segment: DEALER
  Segment: MODEL
  Segment: STOCK
.....
=====
PCB: AUSI2PCB
=====
Segment: SINDXB
.....
=====
PCB: EMPLPCB
=====
Segment: EMPL
  Segment: DEALER
  Segment: SALESINF
  Segment: EMPLINFO
.....

```

Figure 14-8 on page 306 presents the DLIModel diagram. There is one diagram for each of the PCB's in the PSB. As shown in the figure, you can display the details of each of the segments, and on the physical segments you also can make some changes by using the options, available by "right clicking" with the mouse on one of the segments.

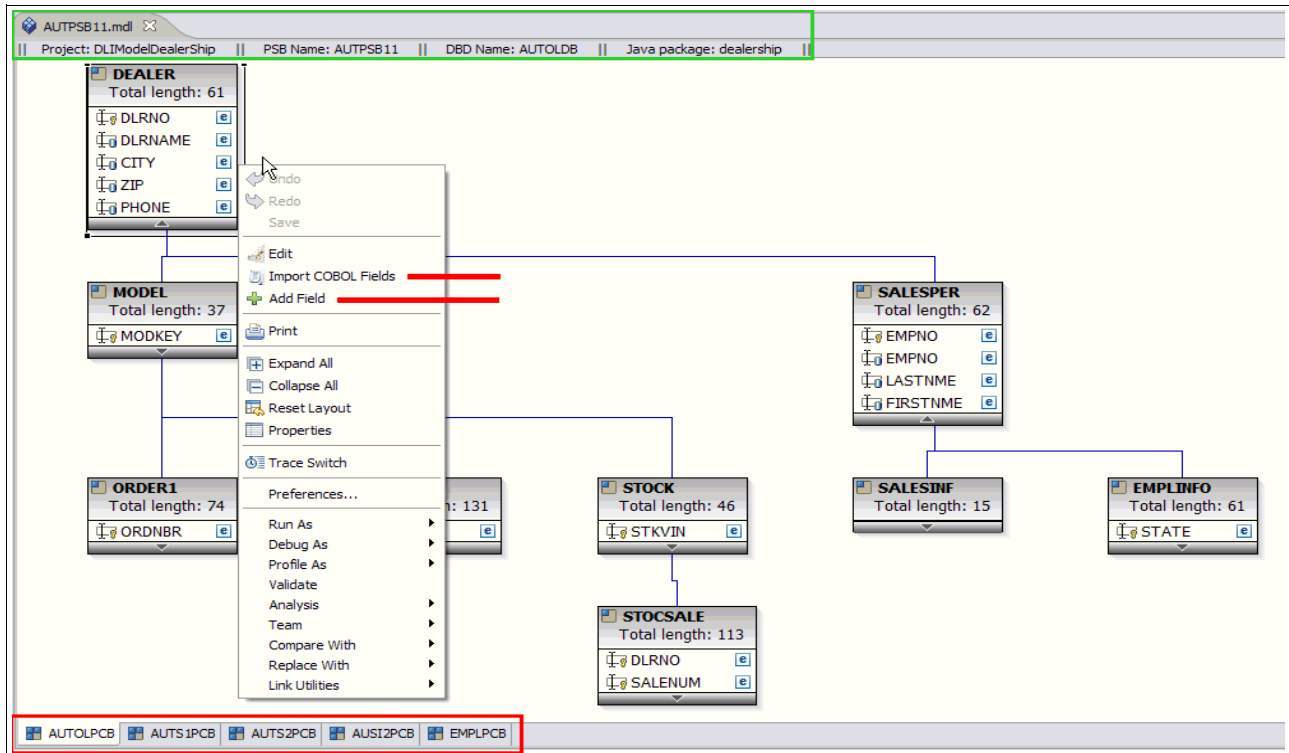


Figure 14-8 Diagram for DealerShip application

14.4 Web Services for DLI databases (IMS V11 and IMS V10 with SPE)

The objective of this enhancement is:

- ▶ To have a Web Services access to DLI data without using the IMS Transaction Manager
- ▶ Provide the industry standard artifacts (WSDL) that can be invoked from any Web Service enabled client.

This enhancement is built on top of the existing and new access facilities for Java to DBCTL thru ODBA.

Figure 14-9 on page 307 illustrates a SOAP Client using the Web Services Description Language (WSDL) to access IMS DB through the web services runtime component, IMS Data Access Services (IMS DAS). IMS DAS is deployed as `imsDAS.jar`, running in WebSphere Application Server z/OS Servant region.

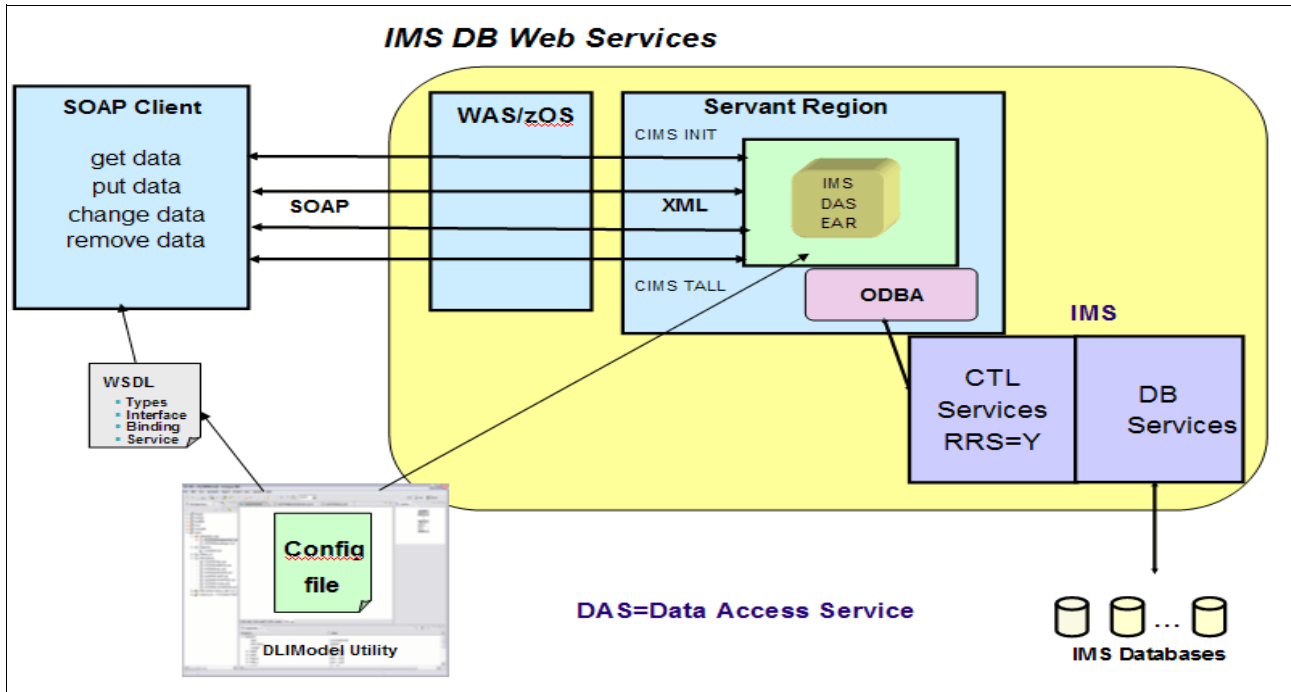


Figure 14-9 Accesses to DLI data from Java APIs

To allow Web Servers access to DLI data, the DLIModel utility must first build WebServices under a DLIModel project. We executed this for an automotive dealer application, and the layout after execution is displayed in Figure 14-10.

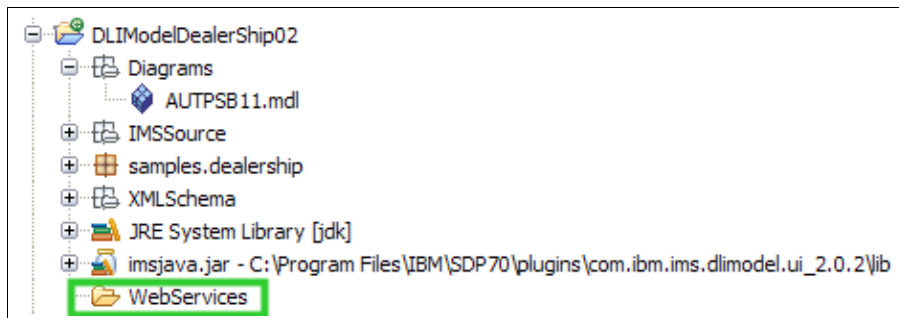


Figure 14-10 DLIModel with Web Services

Using the *WebServices* selection an access configuration is built, which exploits the Program Specification Block (PSB) directives that were reflected in the DLIModelView metadata class.

As an example, we build a *RETRIEVE* access. We build the configuration file by selecting *New IMS DB Web Service* as illustrated in Figure 14-11.

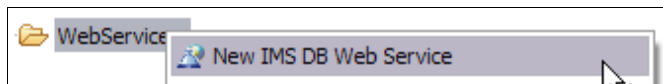


Figure 14-11 Create the Web Services config

In Example 14-2, you find an example of such a configuration file. It is based on the dealership application, on PCB AUTS2PCB of PSB AUTPSB11. The goal is to be able to look up all models associated with a certain dealer.

Example 14-2 WebService configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<Config name="DealerModels" targetNamespace="http://ibm.sj.dealership"
xmlns="http://www.ibm.com/ims/das/config" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<Command name="getDealerModels" kind="RETRIEVE" psb="AUTPSB11" pcb="AUTS2PCB" dbView
="samples.dealership.AUTPSB11DatabaseView" dra="SYS3">
  <Parameter name="DLRNO" type="xsd:string"><xsd:length value="4"/></Parameter>
  <DLI occurrences="10">
    <SSA name="DEALER" >
      <qualifier field="DLRNO" operator="EQ">
        <param name="DLRNO"/>
      </qualifier>
    </SSA>
    <SSA name="MODEL">
    </SSA>
  </DLI>
</Command>
</Config>
```

The configuration file can support the four basic CRUD (Create, Retrieve, Update(Change), Delete) actions. Note in the command line the references to:

- ▶ PSB
- ▶ Program Communication Block(PCB)
- ▶ The name of the java meta class (dbview), generated by the DLIModel utility
- ▶ Data Resource Adapter Name (dra)

As displayed in Example 14-2 the DLI call is expressed as a:

RETRIEVE call on AUTPSB1 (AUTS2PCB), using the AUTPSB11DatabaseView class

In Figure 14-12, the layout of the PCB related to this retrieval of the DLI segments is presented.

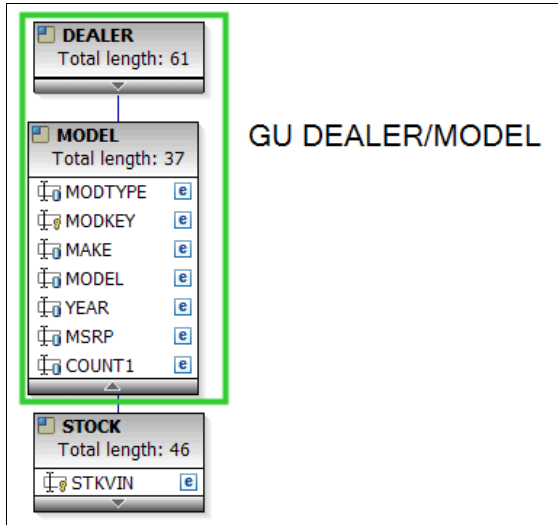


Figure 14-12 PCB used under the PSB

After the configuration is created, we can build the Web Service. All required Java EE artifacts are immediately ready to be deployed as illustrated in Figure 14-13.

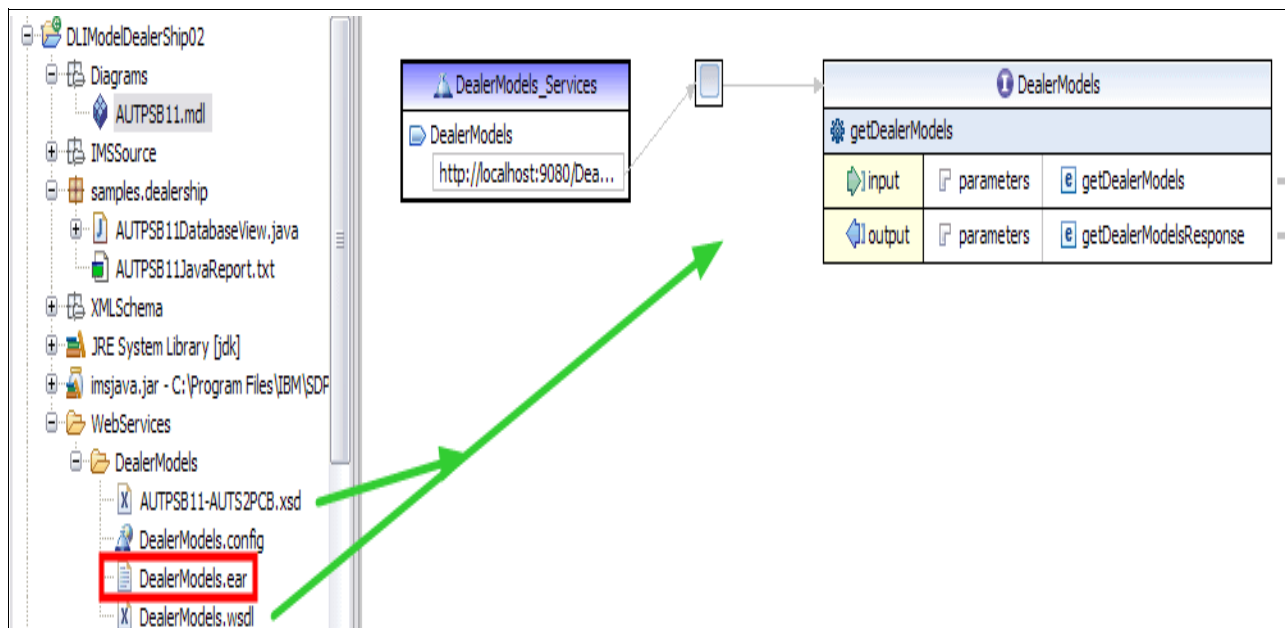


Figure 14-13 Generated elements by built of configuration file

Under the WebServices folder the following are displayed:

- DealerModels.wsdl, which describes the Web service
- DealerModels.ear, the deployable application
- AUTPSB11-AUTS2PCB.xsd, schemas for the message layout

The WSDL contains several sections with a reference to AUTPSB11-AUTS2PCB.xsd, which contains the details about the messages (DLI segment content). The service point of the WSDL is shown below in Example 14-3. If you are interested in the full WSDL code, refer to Example A-6 on page 364. For a listing of the XSD files AUTPSB11-AUTS2PCB.xsd refer to “XSD files for a DLIModel generated Web Services Service” on page 366.

Example 14-3 Service point of the Web service

```

<!--Services-->
<wsdl:service name="DealerModels_Services">
<wsdl:port name="DealerModels" binding="DealerModelsSoapBinding">
<wsdlsoap:address location="http://localhost:9083/DealerModels/services/DealerModels"/>
</wsdl:port>
</wsdl:service>

```

The Enterprise Archive (EAR) is also located under a WebServices directory and is displayed in Figure 14-14 on page 310.

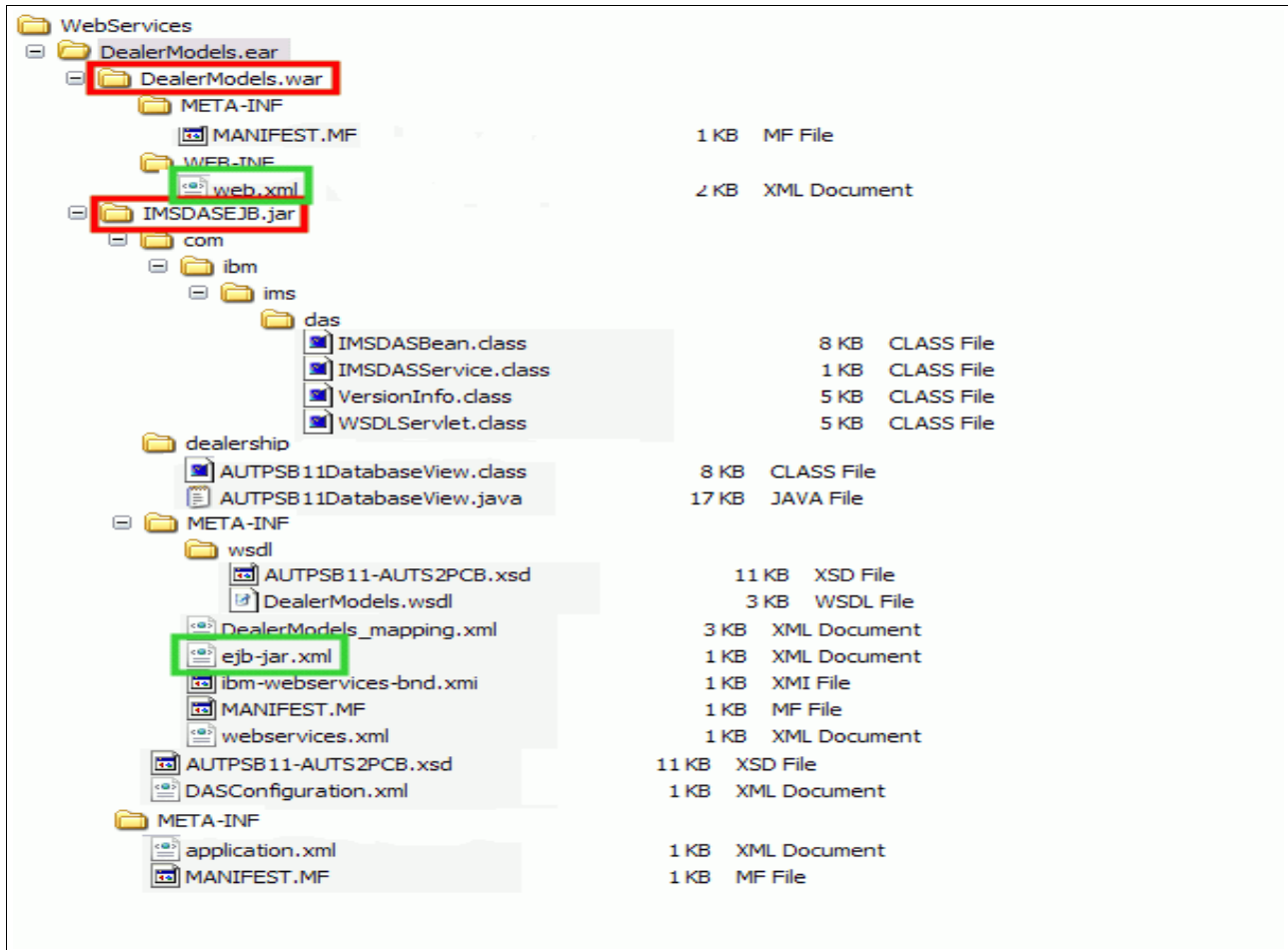


Figure 14-14 EAR file contents

The Enterprise Application Archive, contains a Web application (WAR file) and an Enterprise Java Bean project (JAR file). The WAR part provides the Web Service HTTP service point for the SOAP protocol.

This EAR can be deployed directly in a WebSphere Application Server.

Rational Application Developer offers a few other facilities to test and to use this Web service:

- ▶ The Web Service can be tested from RAD with the Web Services Test Explorer
- ▶ A Proxy can be built for a Java Client

Simply deploy the Web Service in the imbedded WebSphere 6.1 Application Server and enter the following URL to access the Dealer Models sample, which results in output as displayed in Example 14-4.

http://localhost:9083/DealerModels/services/DealerModels

Example 14-4 Output

```
{http://ibm.sj.dealership}DealerModels
Hi there, this is a Web service!
```

Figure 14-15 on page 311 lists the prerequisites for IMS Version 10 Clients to utilize DB Web Services.

Prerequisites

- Software requirements
 - IMS DB Web Services SPE PK73190
 - IMS 10 DB
 - Websphere z/OS 6.1
 - Runtime
 - Admin console to deploy EAR, [imsDAS.jar](#)
 - IBM SDK for z/OS, Java 2 Technology Edition, 5.0 or later
- IMS DB Web Services has dependencies on:
 - IMS Java libraries
 - IMS XML Database
 - return data in XML
- Tooling
 - DLI Model Utility 2.0.2 GUI
 - IMS 10 is the last release to support the z/OS-based batch

Figure 14-15 DB Web Services Version 10 requirements

14.5 IMS, XML, and XQuery

An increasing amount of information is transferred inside and between organizations in the form of XML documents. IMS databases are ideally suited to the storage and retrieval of information in XML documents, given that both are structured hierarchically. Figure 14-16 illustrates this support design.

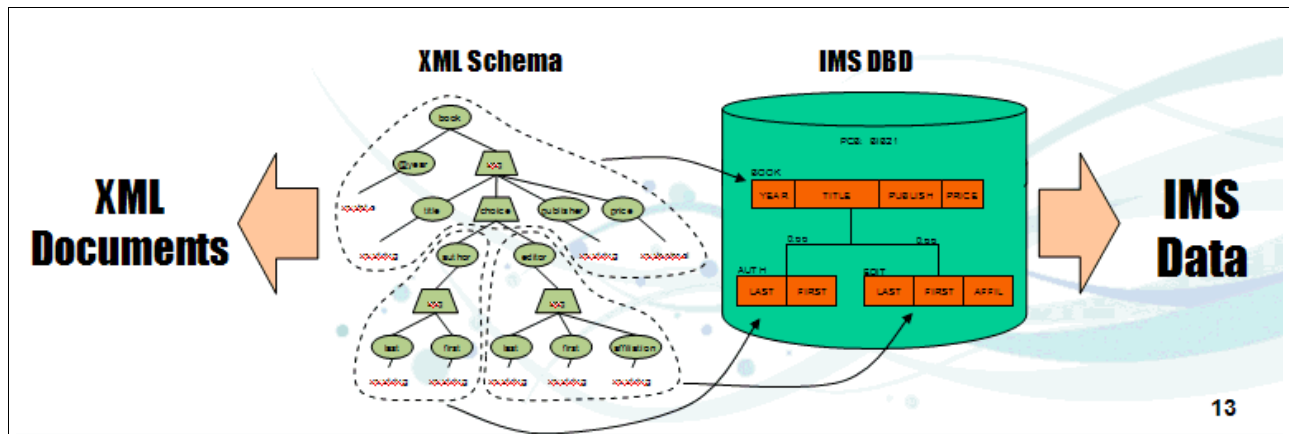


Figure 14-16 DLI databases naturally ready for XML

There are two ways of storing XML documents in IMS databases:

► Decomposed storage

The use of decomposed storage is a data-centric approach to storing the information in IMS database. In this case, the XML data is converted to traditional IMS data types, when stored into the database. Because the data is not stored as XML, existing IMS applications might read and update the information in these IMS databases. For more information, refer to “Decomposed storage mode for XML” on page 313.

► Intact storage

Using intact storage is a document-centric approach to storing the information into your IMS database. In this case, the XML data is not converted when stored in an IMS database. The data is stored as XML, and includes all the XML tags and the information. This data cannot be manipulated by non-XML applications. For more information refer to “Intact storage mode for XML” on page 314.

IMS also provides tools to assist you to define the metadata for mappings between your IMS database definitions and those required for XML. This is provided through an extension of the DLIModel utility.

This utility reads PSB and DBD source, XML Metadata Interchange Format (XMI) files, and control statements. The XML schema is used by an IMS Java application to compose a XML document from an IMS database record, or to compose IMS database segments from a XML document. Figure 14-17 provides an overview of this use of the DLIModel utility.

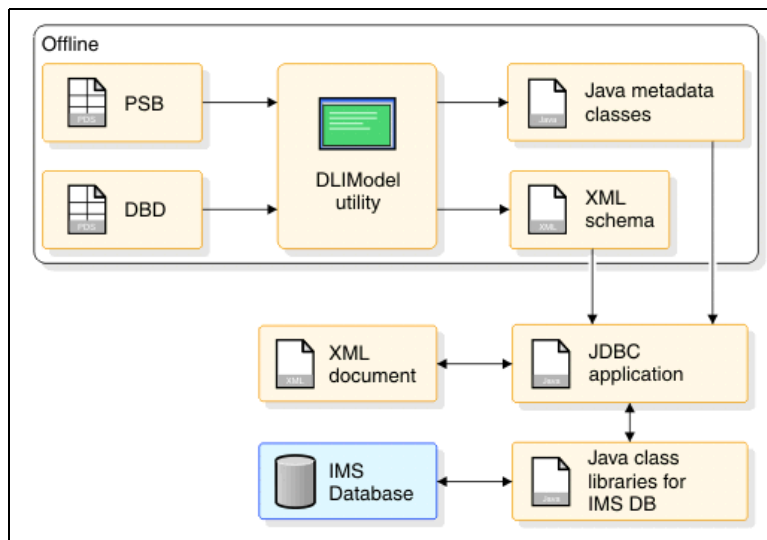


Figure 14-17 The DLIModel utility and XML

A XML schema defines what data might be present in a XML document. The schema can be used to create a XML document from IMS database data by describing how to copy data from the segments into a XML document.

14.5.1 XML storage in IMS databases

IMS allows you to easily receive and store incoming XML documents and compose XML documents from existing information stored in IMS databases.

Figure 14-18 on page 313 presents three aspects of the storing of a XML documents in DLI:

- IO aspect: Is controlled by the description of the DLI database and the selected DLI organization (HDAM, HIDAM, HALDB...) and the presence of Secondary indexes. This is in the database descriptor.
- Hierarchical Model: Requires the DBD and additional DLI Meta information provided by the DLIModel utility.
- XQuery/XPath: Is exploitable through IMS Version 10 onward, providing an enhanced RetrieveXML function.

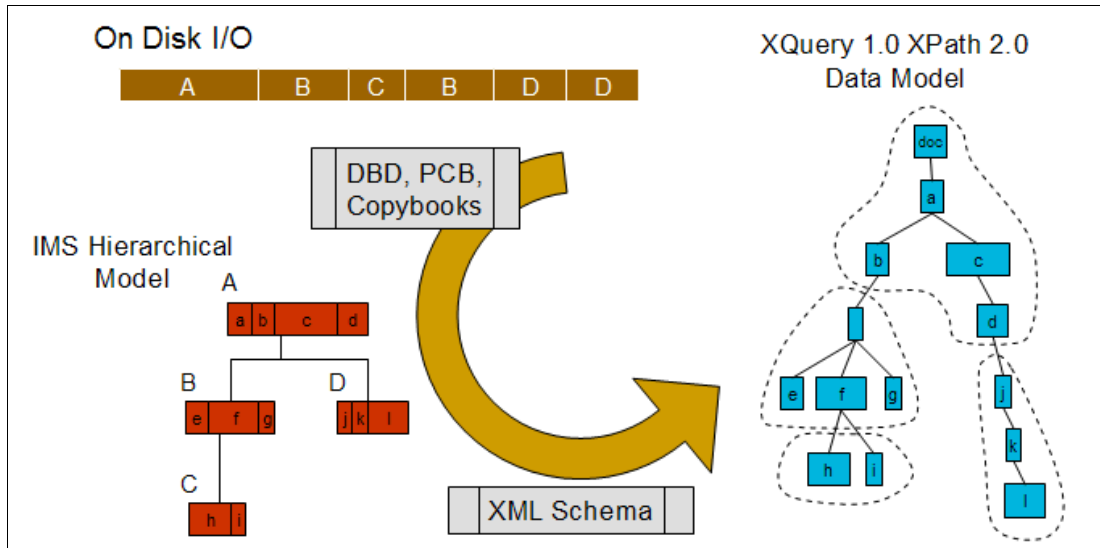


Figure 14-18 Three aspects of storing XML documents in DLI

You can store XML documents decomposed, intact, or in a combination of decomposed and intact.

Decomposed storage mode for XML

In decomposed storage mode, all elements and attributes are stored as regular fields in optionally repeating DL/I segments. During parsing, all tags and other XML syntactic information is checked for validity and then discarded. The parsed data is physically stored in the database as standard IMS data, meaning that each defined field in the segment is of an IMS standard type. Because all XML data is composed of string types (typically Unicode) with type information existing in the validating XML schema, each parsed data element and attribute can be converted to the corresponding IMS standard field value and stored into the target database. Figure 14-19 provides a pictorial representation of this.

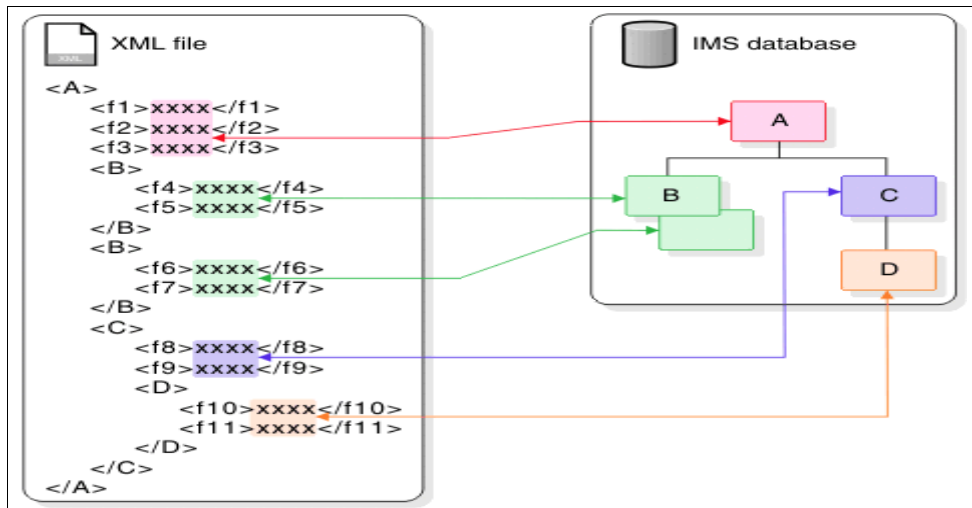


Figure 14-19 How XML is decomposed and stored in IMS segments

Inversely during XML retrieval, DL/I segments are retrieved and fields are converted to the destination XML encoding, tags and XML syntactic information (stored in the XML schema) are added, and the XML document is composed.

Decomposed storage mode is suitable for data-centric XML documents, where the elements and attributes from the document typically are either character or numeric items of known short or medium length that lend themselves to mapping to fields in segments. Lengths are typically, though not always, fixed.

The XML document data can start at any segment in the hierarchy, which is the root element in the XML document. The segments in the subtree below this segment are also included in the XML document. Elements and attributes of the XML document are stored in the dependent segments of the root element segment. Any other segments in the hierarchy that are not dependent segments of that root element segment are not part of the XML document and, therefore, are not described in the describing XML schema.

The XML hierarchy is defined by a PCB hierarchy that is based on either a physical or a logical database. Logical relationships are supported for retrieval and composition of XML documents, but not for inserting documents.

Intact storage mode for XML

In intact storage mode, all or part of a XML document is stored intact in a field. The XML tags are not removed and IMS does not parse the document. XML documents can be large, so the documents can span the primary intact field which contains the XML root element, and fields in overflow segments. The segments that contain the intact XML documents are standard IMS segments, and can be processed like any other IMS segments. The fields, because they contain unparsed XML data, cannot be processed like standard IMS fields. However, intact storage of documents have the following advantages over decomposed storage mode:

- ▶ IMS does not need to compose or decompose the XML during storage and retrieval. Therefore, you can process intact XML documents faster than decomposed XML documents.
- ▶ You do not need to match the XML document content with IMS field data types or lengths. Therefore, you can store XML documents with different structure, content, and length within the same IMS database.

Intact XML storage requires a new IMS database or an extension of an existing database because the XML document must be stored in segments and fields that are specifically tailored for storing intact XML.

To store all or part of a XML document intact in an IMS database, the database must define a base segment which contains the root element of the intact XML sub-tree. The rest of the intact XML sub-tree is stored in overflow segments, which are child segments of the base.

IMS cannot search intact XML documents for specific elements within the document. However, you can create a side segment that contains specific XML element data. This segment can then be searched with a secondary index. Figure 14-20 on page 315 presents this design.

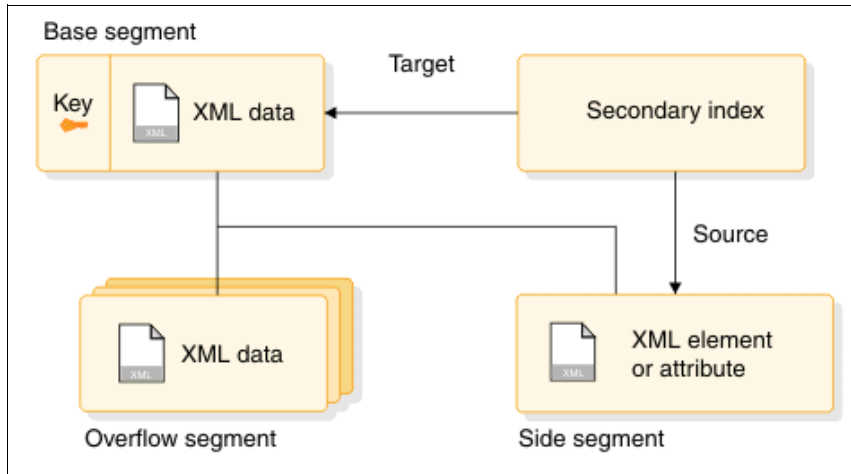


Figure 14-20 Intact storage of XML with a secondary index

For more info about this subject, refer to *IMS Database Administration Guide version 10*, SC18-9704.

14.5.2 SQL extensions for XML storage and retrieval

The Java class libraries for IMS have two SQL99 extensions for user-defined functions (UDFs):

- ▶ retrieveXML
- ▶ storeXML

These UDFs are used during JDBC calls to store and retrieve XML from IMS databases. This interface is independent of the physical storage of the data.

retrieveXML UDF

The retrieveXML UDF creates a XML document from an IMS database and returns an object that implements the *java.sql.Clob* interface. It does not matter to the application whether the data is decomposed into standard IMS segments or the data is in intact XML documents in the IMS database.

JDBC stores a Character Large Object (*java.sql.Clob*) as a column value in a row of the result set. The *getClob()* method retrieves the XML document from the result set. Figure 14-21 on page 316 presents the relationship between the retrieveXML UDF and the *getClob()* method.

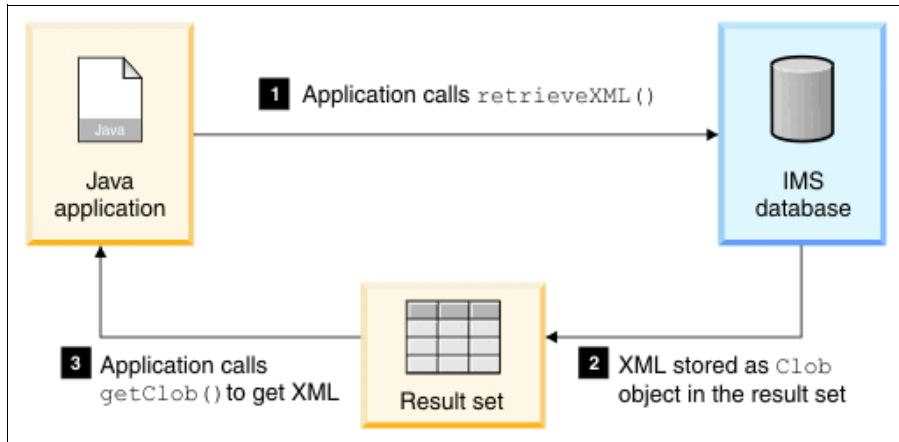


Figure 14-21 retrieveXML

To create a XML document, use a retrieveXML UDF in the SELECT statement of your JDBC call. Pass in the name of the segment that will be the root element of the XML document (for example, retrieveXML(Model)). The dependent segments of the segment that you pass in are in the generated XML document if they match the criteria listed in the WHERE clause.

storeXML UDF

The storeXML UDF inserts a XML document into an IMS database at the position in the database that the WHERE clause indicates. IMS, not the application, uses the XML schema and the Java metadata class to determine the physical storage of the data into the database. It does not matter to the application whether the XML is stored intact or decomposed into standard IMS segments.

A XML document must be valid before in can be stored into a database. The storeXML UDF validates the XML document against the XML schema before storing it. If you know that the XML document is valid and you do not want IMS to revalidate it, use the storeXML(false) UDF.

To store a XML document, use the storeXML UDF in the INSERT INTO clause of a JDBC prepared statement. Within a single application program, you can issue INSERT calls that contain storeXML UDFs against multiple PCBs in an application's PSB.

14.5.3 XQuery

IMS Version 9 introduced a method to start viewing your IMS data (from existing or new IMS databases) as collections of XML documents, by aligning the hierarchical structure of an IMS database (and therefore the IMS records stored within it) with a corresponding hierarchically structured XML Schema (and therefore the XML documents valid to it).

It did not, however, offer a meaningful way to query this new view of a collection of XML documents. To really be useful as a business tool, we need to be able to search, aggregate, evaluate, and essentially pick and choose the parts of our XML collection that are important and then convert that resulting data into XML. This is exactly why IBM, Oracle, Microsoft and many more of the industry database leaders joined together in creating the w3c standard XQuery language. XQuery is a powerful query language that can be thought of as the SQL for hierarchically structured data.

Support in IMS Version 10 for XQuery is the implementation of an XQuery evaluation engine that runs against IMS databases. This support uses the existing IMS-to-XML mapping rules

that are available in IMS Version 9, but extends these mapping rules so that they can more flexibly store and retrieve information. To use XQuery with IMS Version 10, JDK™ 5.0 must be installed and imsXQuery.jar must be included in the application classpath.

In Figure 14-22 we present highlights of XQuery support, plus sample coding and the result set produced.

XQuery support in IMS 10

- Further aligns IMS with industry direction
 - XML, SOA, Web Services, etc.
- More natural fit for hierarchical data querying
- Enables customers to leverage emerging standard skill set
- Enhanced product and tooling integration
- Immediately usable with no migration of existing IMS data

```

<bib> {
  for $b in /bib/book
  let $title := $b/title
  where $b/publisher = "Addison-Wesley"
  order by $b/@year
  return
    <book year="{ $b/@year }">
      { $title }
    </book>
} </bib>

```

➔

```

<bib>
  <book year="1992">
    <title>Advanced Programming in the Unix
  </book>
  <book year="1994">
    <title>TCP/IP Illustrated</title>
  </book>
</bib>

```

Figure 14-22 Overview of XQuery support provided by IMS Version 10

The IMS DB Adapter XQuery support extends the retrieveXML UDF by adding a second parameter. The second parameter allows the passing of an XQuery 1.0 expression. The expression is evaluated relative to the retrieveXML context and returned to the result set as a CLOB value.

For IMS XQuery support, the XQuery 1.0 and XPath 2.0 Data Model serves two purposes:

- ▶ It defines the information contained in the input to be used by the IMS XQuery processor
- ▶ It defines all permissible values of expressions in the XQuery, and XPath languages that can be evaluated by the IMS XQuery processor.

XPath 2.0 is an expression language that allows the processing of values conforming to the data model defined in [XQuery/XPath Data Model (XDM)]. The data model provides a tree representation of XML documents and atomic values such as integers, strings, and booleans, and sequences that might contain both references to nodes in a XML document and atomic values. The result of an XPath expression might be a selection of nodes from the input documents, or an atomic value, or more generally, any sequence allowed by the data model. The name of the language derives from its most distinctive feature, the path expression, which provides a means of hierarchic addressing of the nodes in a XML tree.

A good explanation about the XPath API can be found at:

<http://www.ibm.com/developerworks/library/x-javxpathapi.html>

XPath expressions are much easier to write than detailed Document Object Model (DOM) navigation code. When you need to extract information from a XML document, the quickest and simplest way is to use an XPath expression, based on the FLWOR pattern.

The name FLWOR comes from the five clauses that make up a FLWOR expression:

- for*: iterates through a sequence, bind variable to items
- let*: binds a variable to a sequence
- where*: eliminates items of the iteration
- order by*: reorders items of the iteration
- return*: constructs query results

To see how FLWOR expressions work, just look to the explanation of the logic behind the XQUERY, as shown in Figure 14-23.

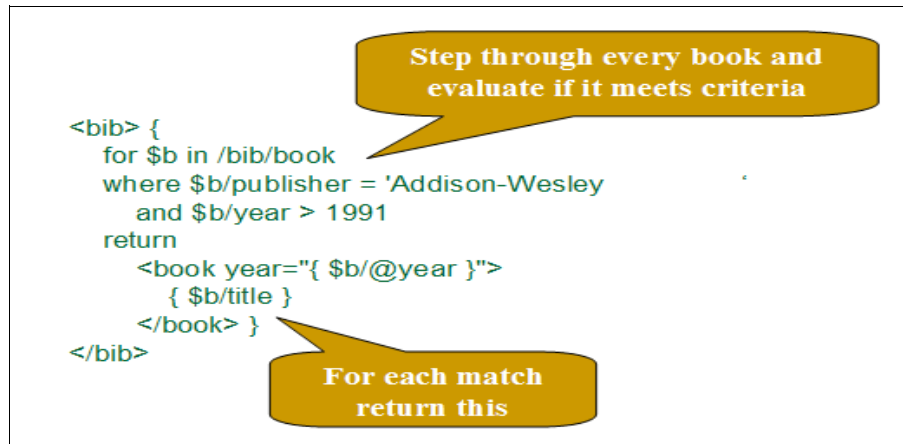


Figure 14-23 What happens in the XQuery

You can optimize your search results with XQuery support with the JDBC driver for IMS by further refining your results with IMS-specific function and operation extensions as presented in Figure 14-24.

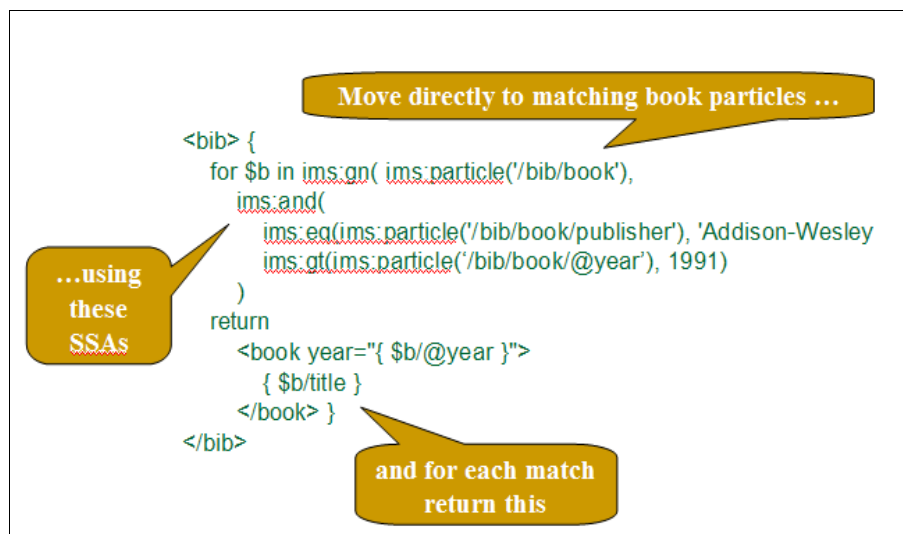


Figure 14-24 XQuery adapted for IMS DLI language

Refer to the following manual for more information about this subject, read *IMS Application Programming Guide Version 10*, SC18-9698.

14.6 Conclusion

ODBA makes it possible to have direct DLI access to IMS Databases from a non IMS environment.

In the Java language, a JDBC type 2.1 access is possible after generating the meta representation of the databases corresponding to a particular PSB. A lower level DLI API also exists. Those facilities can be used locally, but also from a distributed platform through IMS Connect. From the distributed platform we can use a JDBC type 4 access for DLI.

Other meta data produced by the DLIModel utility provides the support for the XML exploitation of the hierarchical organized DLI data. The DLIModel was extended so that a Web Service can be generated for a PSB.



IMS Open Database and Universal Drivers

In this chapter, we provide information about two exciting new IMS components: IMS Open Database and IMS Universal Drivers

IMS Open Database:

- ▶ Supports open standards that can be used for distributed and local connectivity to IMS databases.
- ▶ Manages access to databases that IMS DB systems own.

IMS Universal Drivers release a flexible way to access your existing and new IMS database resources locally and remotely. IMS Universal Drivers consist of three distinct parts:

- ▶ A DB resource adapter
- ▶ A DL/I driver
- ▶ A JDBC driver

With these drivers, you can write new applications and enhancements to existing IMS applications using a Java API that includes DL/I and JDBC SQL interactions. The DB resource adapter uses the standard Common Client Interface (CCI) and a JDBC Interface to provide access to IMS from both a standalone Java SE and a Java EE Server.

Historically, IMS database was closed architecture, and by opening it up, IMS is positioned for the future as it pertains to industry-standard access APIs and the emerging SOA market.

15.1 IMS Open Database

IMS Open Database support enhances the distributed access to IMS data and the availability of IMS. There are important benefits that are associated with the use of IMS Open Database:

- ▶ Allows IMS databases to be processed as a standards-based data servers.
- ▶ Access to IMS databases can occur across LPAR boundaries and across network boundaries.
- ▶ Protects IMS control regions from the unexpected termination of the application program by a reduction on the impact of an ODBA client failure that can result in an IMS U113 abend.

Three main components combine to provide Open Database access to IMS Databases:

- ▶ The Open Database Manager
- ▶ Modifications to IMS Connect to support TCP/IP access to the Open Database Manager environment. Industry standard Distributed Relational Database Architecture (DRDA) protocols are used to communicate with IMS Connect. IMS Connect becomes the gateway to IMS transactions and data.
- ▶ The Open Database APIs that can be used to access IMS DB through the Open Database Manager environment. With IMS 11, we are rolling out a complete suite of Universal drivers in support of IMS database connectivity and programmatic access. The intent is to access IMS in a uniform way, using the most relevant industry standards from any platform.

We start our discussion by defining terms that are associated with IMS Open Database:

IMSplex	One or more IMS systems working together as a single unit.
CSL	Common Service Layer is the infrastructure needed for IMSplex systems and resource management.
CCTL	Coordinator Controllers are transaction programs that use the DRA to access online IMS databases.
DRA	The Database Resource Adapter is an interface to IMS HALDB, full-function, and DEDB databases. Do not confuse this with the Java EE DB Resource Adapter, which is the interface the Open DB drivers provides for Java EE component access to IMS databases.
DRA Thread	Structure connecting a CCTL task or an ODBA task with an IMS DB task.
ODBA	Open Database Access are application programs that use the DRA to access online IMS databases.
ODBM	The Open Database Manager is a CSL address space that manages connections to online IMS databases.
ODBM Client	Programs that access ODBM such as IMS Connect.
Open Database API	Provides Universal Drivers for direct access to IMS databases.
Type 4 TCP/IP Connection	Network and cross LPAR distributed environments for IMS database access.
Universal Drivers	Supports Type 4 connections for Java application programs.

15.1.1 Review of ODBA use

Open Database Access (ODBA) is a callable interface to access databases that the IMS Database Manager manages. It has been available since IMS Version 7. Any z/OS application program that uses the Recovery Resource Services (RRS) of z/OS can use ODBA as a sync point manager to access the IMS full function databases and DEDBs. Figure 15-1 displays existing ODBA support.

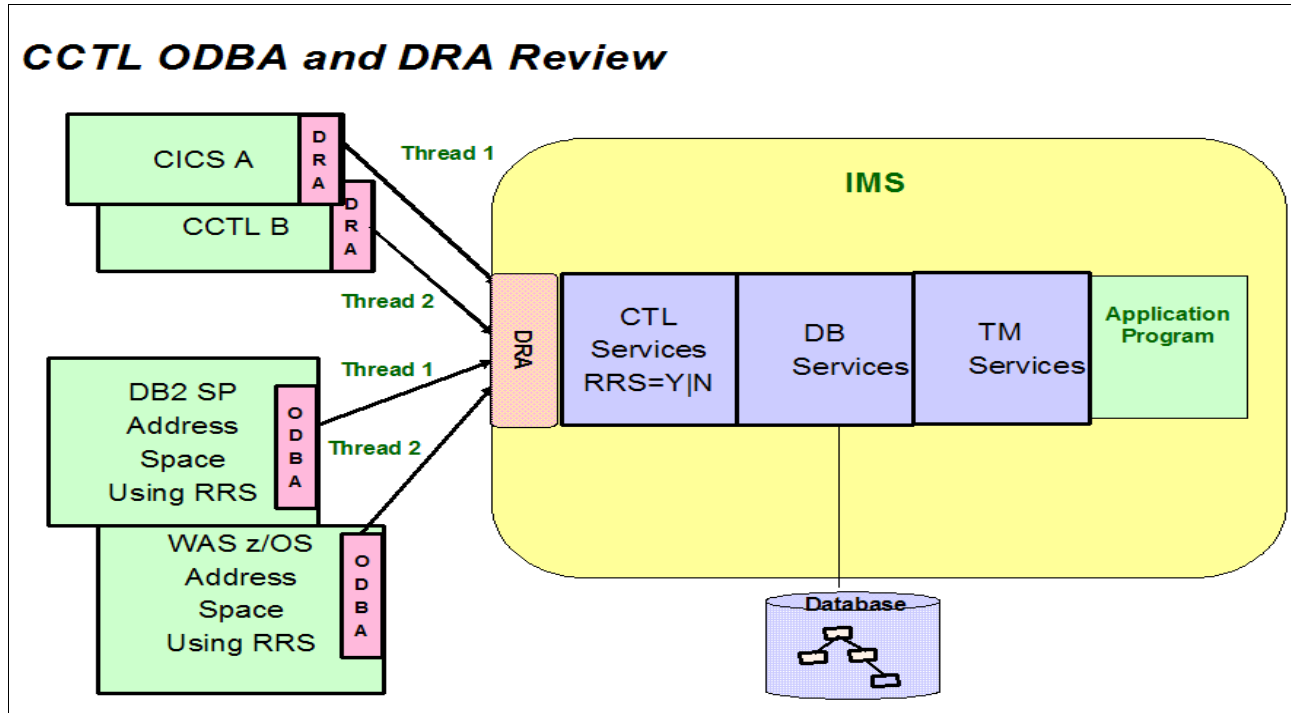


Figure 15-1 Review of existing ODBA and DRA architecture

The Database Resource Adapter (DRA) is an interface to IMS DB full function databases and DEDBs. The DRA can be used by a coordinator controller (CCTL) or a z/OS application program that uses the Open Database Access (ODBA) interface. z/OS ODBA application programs issue DL/I calls using an application interface block (AIB).

This solution leverages ODBA as the API to access IMS database resources. ODBA can make address space to address space calls (PC calls) in the same logical partition. The net effect of this is that the ODBA modules must be on the same LPAR as the IMS CTL region. The ODBA modules are loaded in the address space of the application, which is in turn is loaded in the address space of the container. In this case, the container is WebSphere Application Server, and as a result of this the WebSphere Application Server installation must be on the same LPAR as the IMS DB itself.

This restriction was resolved through the use of Open Database, which we introduce now.

15.1.2 Open Database Manager

Figure 15-2 on page 324 provides an overview of the Type 2 and 4 connections used by the Universal Drivers and the use of the Open Database Manager address space. The Open Database Manager (ODBM) is a CSL address space and supports SCI API for ODBM Clients; such as IMS Connect, access to ODBM. What we are doing is creating a new CSL address space to house the ODBA modules. This interface uses SCI as its communication

mechanism. The ODBA modules are no longer tightly coupled with the applications themselves, and therefore the containers.

SCI uses either PC or XCF calls to communicate with other SCI components and XCF allows calls to go across LPARs in an IMSplex, which allows applications and their containers to be isolated on their own LPARs. The ODBM address space must be on the same LPAR with IMS due to the use of the ODBA interface.

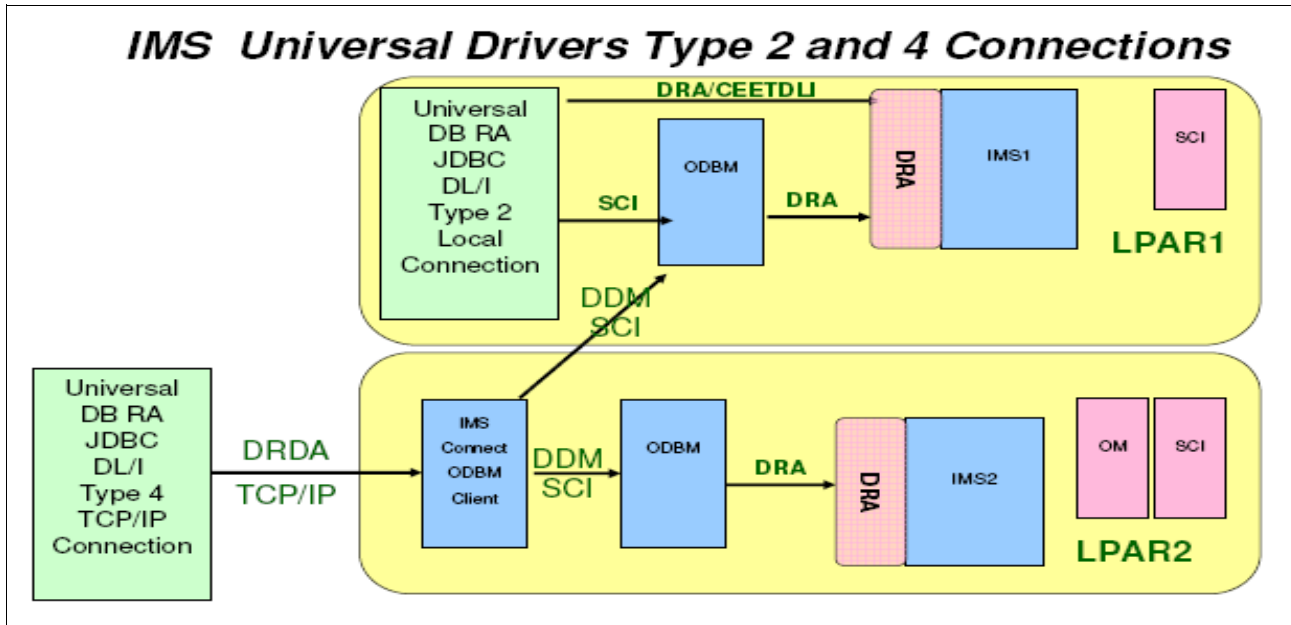


Figure 15-2 Use of the IMS Universal Drivers with Open database concepts

Figure 15-3 on page 325 presents the Open Database environment where no WebSphere on the host is required and an IMS Connect image on one LPAR can front end the requests for DL/I services to multiple ODBMs.

ODBM uses the Database Resource Adapter (DRA) interface to access IMS. Because IMS DB is a participant in the two phase commit process, ODBM can provide the CCTL sync point coordinator function and can use RRS as the sync point coordinator for the ODBA interface.

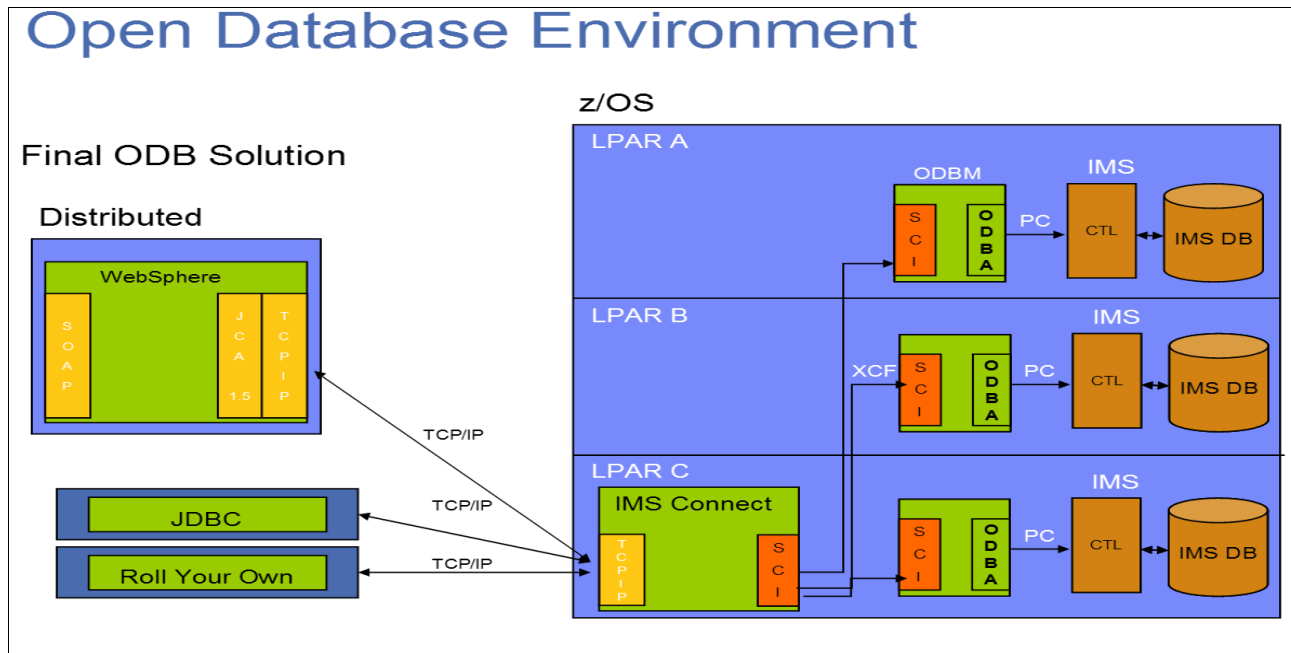


Figure 15-3 The target environment using Open Database

Our goal is to leverage IMS Connect as the complete gateway solution for IMS TM, OM, and now DB. IMS Connect will be augmented in IMS Version 11 to be an ODBM client, which allows distributed applications to leverage the TCP/IP protocol to communicate with IMS Connect, which can then access any database in the entire IMSplex.

The key benefit is that distributed clients would now have the option of going directly to IMS Connect for IMS DB requests. Existing DB Resource Adapter applications are unaffected by Open Database. To exploit Open Database from existing DB Resource Adapter applications, a migration to the JCA 1.5 programming model must be performed.

Before establishing the connection to the IMS system, ODBM translates the incoming database requests from the DRDA protocol (submitted by the IMS provided connectors and user-written applications) into the DL/I calls expected by IMS. DRDA is built on top of DDM, which is a lower level protocol. When ODBM returns the IMS output to the client, ODBM translates the response to the DRDA protocol.

15.2 IMS Database Resource Adapters and the IMS Universal Drivers

In this section, we compare the IMS database resource adapters to the IMS Universal Drivers architecture and deployment. First, a few comments on where these technologies intersect.

The IMS Universal Drivers support the functionality of all current IMS database resource adapters. Both the IMS database resource adapters and IMS Universal Drivers include three components:

- ▶ A DB resource adapter, which is a Java EE installable resource adapter that can be deployed on a Java EE within a server (for example, WebSphere Application Server).
- ▶ A JDBC driver, which is a Java application that supports the SQL interactions.
- ▶ A DL/I driver, which is a Java application that supports the DL/I interaction.

The current JDBC and DL/I drivers are deployed in IMS, CICS, or DB2 stored procedures. They support a subset of SQL commands and IMS Type 2 commands. Place these drivers in the IMS Java message processing (JMP) or Java batch processing (JBP) region. Figure 15-4 is an overview of how IMS-dependent regions that execute Java code can access IMS databases, and the current interfaces supporting access to IMS data stores from external callers.

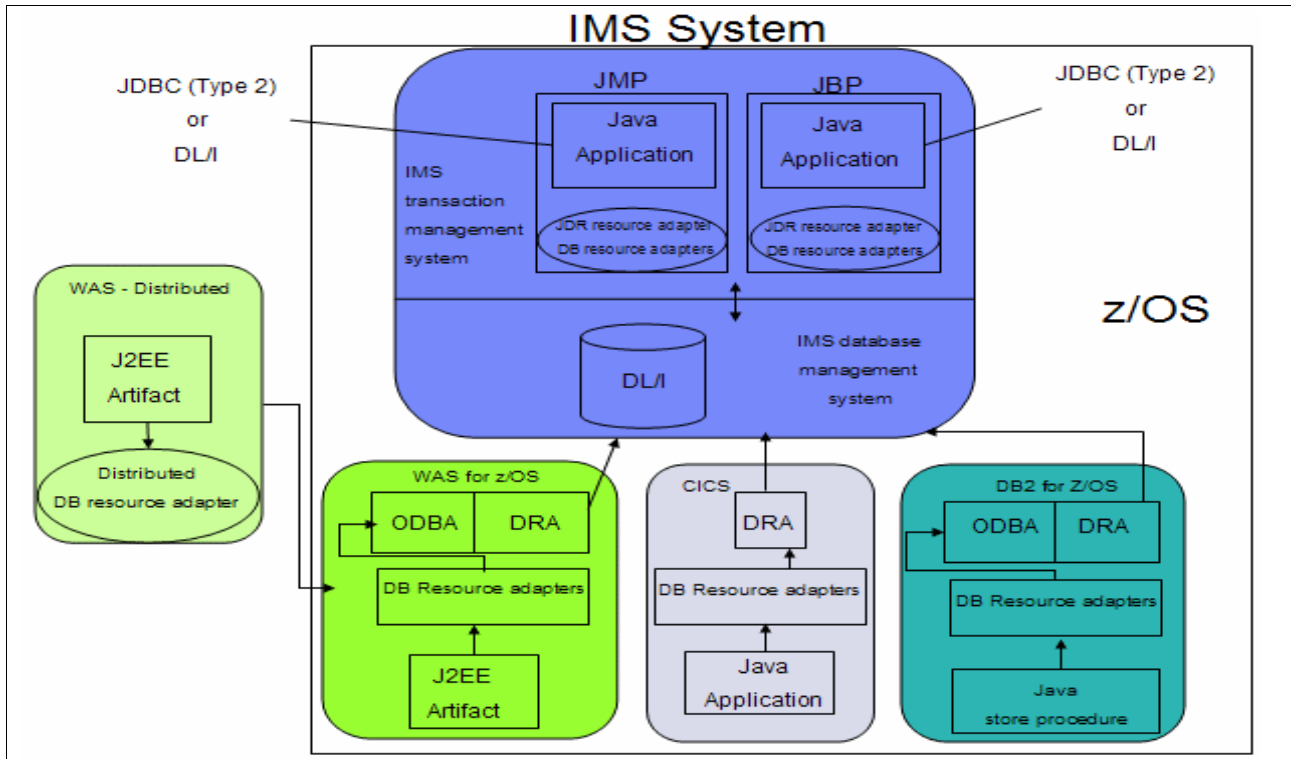


Figure 15-4 Current Type 2 support for the Database resource adapters

In IMS Version 11, the JDBC driver Type 4 and type 2 are supported, the driver Type 4 support allows access to an IMS database remotely, which eliminates the limitation of it being necessary to deploy JDBC or DL/I Java applications in the Java-dependent region locally only; however, that option is still valid when it is needed. These new features make IMS database an open architecture, which takes away the requirement of having a z/WebSphere Application Server installed in a z/OS environment when using a distributed WebSphere Application Server to get access to IMS data remotely. Figure 15-5 on page 327 displays Type 2 support with the Universal Drivers.

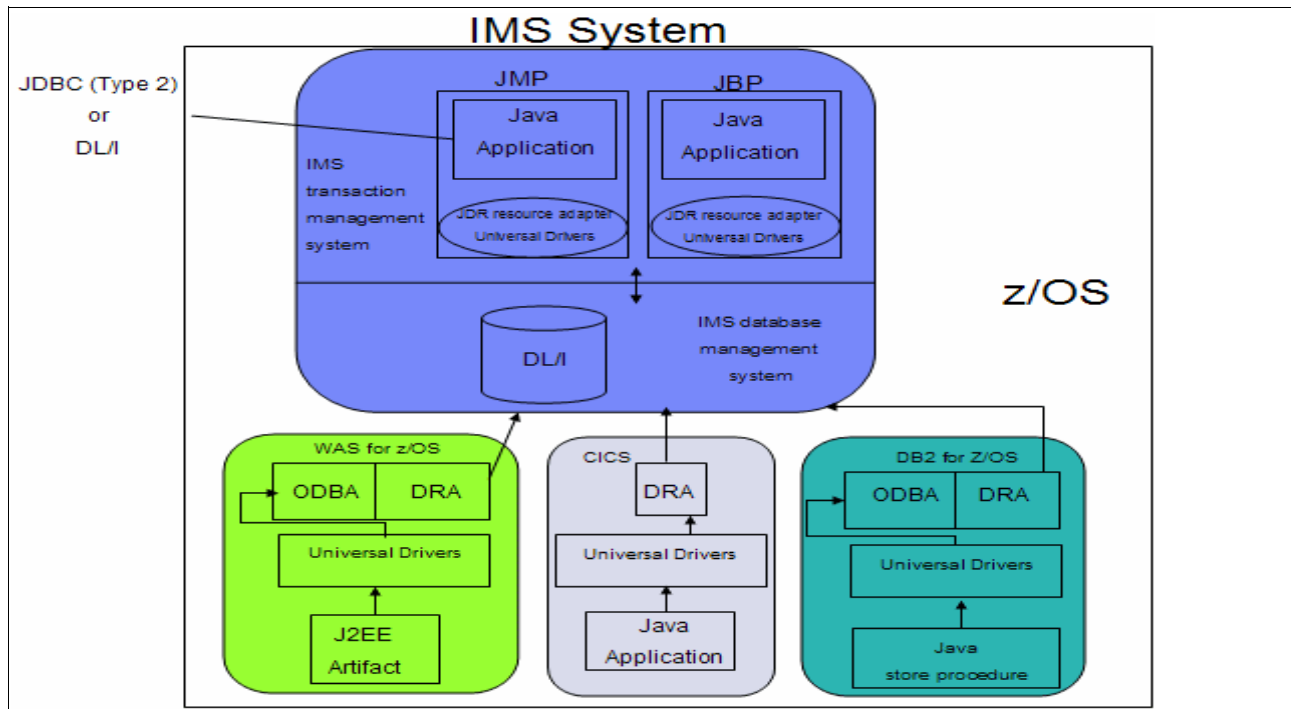


Figure 15-5 Universal Drivers Type 2 support

The new DB resource adapter that is shipped with the universal drivers is JCA 1.5 compliant and provides the ability to access IMS databases from z/OS platforms and non z/OS platforms without writing an IMS transaction. JCA has two layers that are supported with the new drivers:

- ▶ The Service Provider Interface (SPI)

The SPI system contract defines the types of interactions that take place between a resource adapter and the various qualities of services that the Java EE container provides, such as connection, security, and transaction management.

- ▶ Common Client Interface (CCI)

The Common Client Interface (CCI) governs the interactions between an EJB, which contains an application's business logic, and a JCA resource adapter. It is not necessary to make any changes to existing Java EE artifacts (for example, EJB) that were written with the current JCA 1.0 DB resource adapter because the JCA 1.5 also supports the back level version.

With this new support, it is not necessary to have a Java application loaded into the IMS Java dependent region to access IMS databases. In addition, it is not necessary to have two WebSphere Application Servers to communicate to IMS databases. In Figure 15-6 on page 328, the distributed WebSphere Application Server uses Internet Inter-ORB Protocol (IIOP) to communicate to the z/OS WebSphere Application Server to access DL/I data. The IMS Version 11 DB resource adapter does not have this limitation. It only needs to be deployed on a distributed WebSphere Application Server that uses TCP/IP to communicate to IMS Connect to access DL/I data.

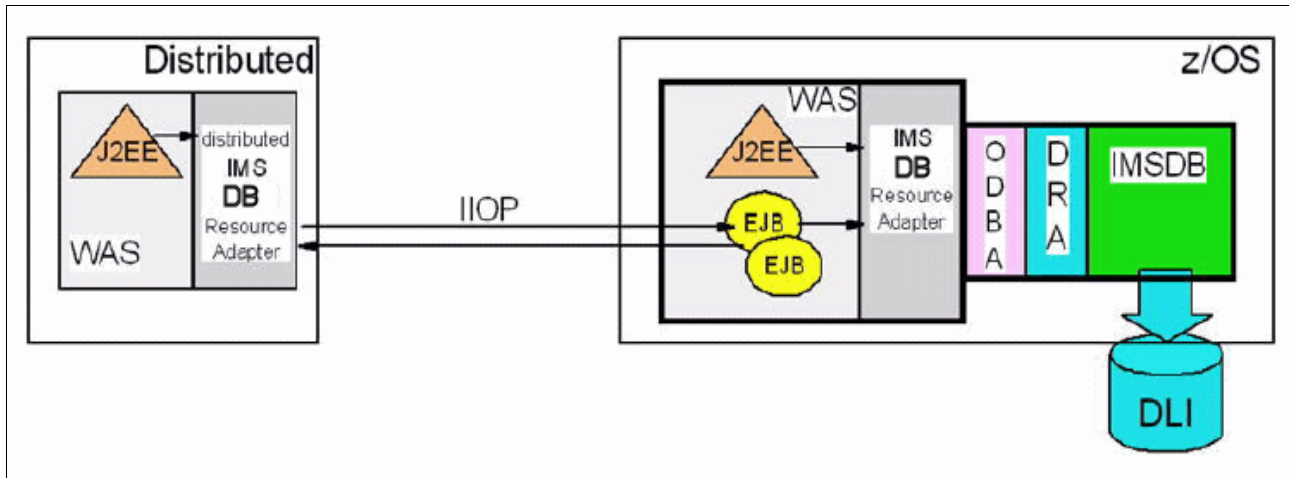


Figure 15-6 Accessing DLI data with the IMS DBRA in IMS Version 10 and earlier

ODBA can make cross address space calls (PC calls) but only within the same logical partition, which means that the application and ODBA modules need to be on the same LPAR that the IMS CTL region is on. The ODBA modules are loaded into the address space of the application, which is running under its container's TCB, which poses a potential problem because if WebSphere Application Server (with ODBA within it) is terminated while an application's DLI call is in process, then the IMS control region receives an abend U113 when it detects that the ODBA TCB is no longer around.

Another obvious problem here is that this solution is not scalable. As your company moves toward implementing a service-oriented architecture, it is unrealistic to have to install and maintain a WebSphere Application Server on every LPAR where you have IMS data. Both of these problems now have a solution in the IMS Universal Drivers.

The IMS Universal Drivers will eventually phase out the previous versions of the IMS database resource adapters. In summary, IMS Universal Drivers will encompass all current Database resource adapters features and provide enhancements that can be used to access your IMS data using Java applications remotely through a Java EE server (managed and non-managed) or as stand-alone Java SE applications.

15.3 Overview of how the IMS Universal drivers work with IMS

Figure 15-7 on page 329 shows the new DB resource adapter deployed on a distributed WebSphere Application Server accessing multiple IMS databases from a single Java EE artifact.

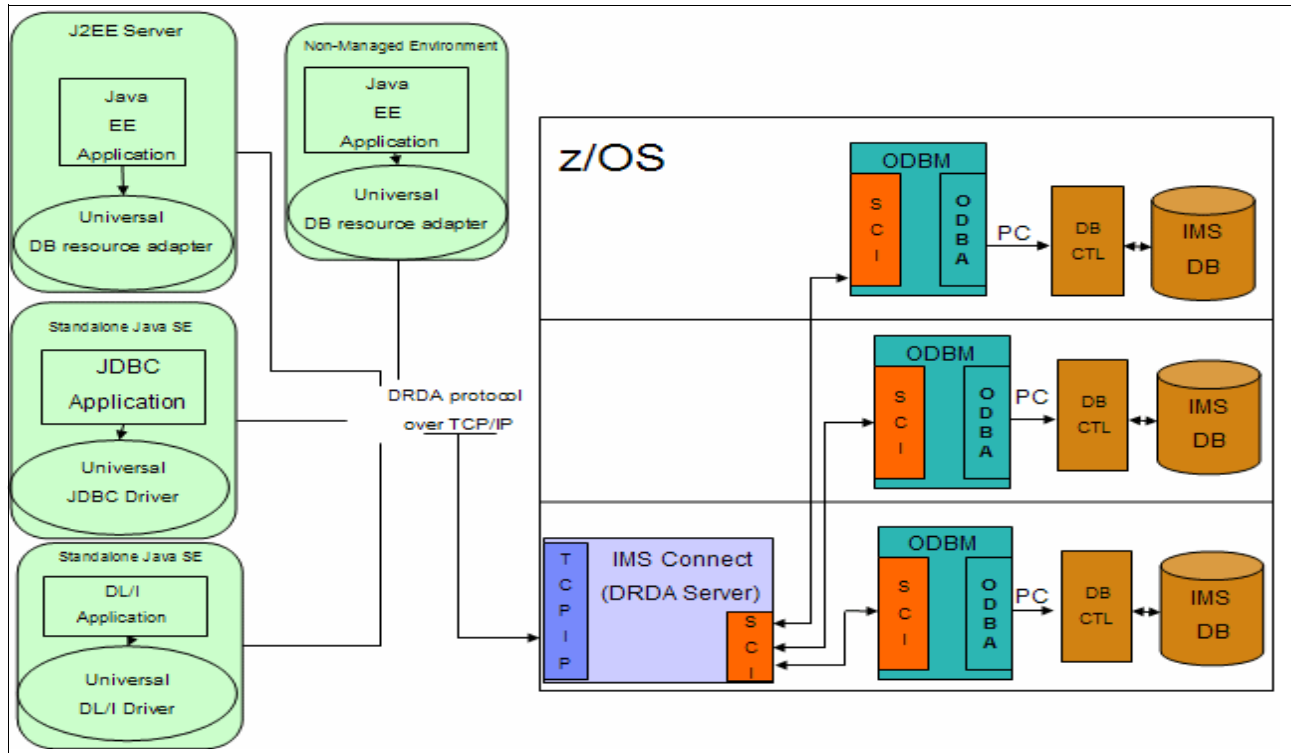


Figure 15-7 IMS Universal Drivers remote connectivity across multiple LPARs

Now that we compared the current drivers and new drivers, you might wonder how does this all perform. There are three main components that make it possible for IMS database to become an open architecture:

- ▶ Complete client-side libraries, which implement the industry standards interfaces and protocols.
- ▶ IMS Connect, which routes requests for database resources to the proper ODBM and routes responses back.
- ▶ The Open Database Manager address space, which is vital in the processing of all database access requests.

The IMS Universal drivers use IMS Connect and Open Database Manager to provide true distributed access. The IMS Universal drivers broker all communications between your Java applications and the IMS subsystem by sending DRDA messages over the TCP/IP protocol. IMS Connect serves as the TCP/IP server or router. IMS Connect uses an internal routing table to find the ODBM that supports the IMS alias that is requested by the client and passes the request to ODBM. ODBM is a separate address space that is running under the IMS Common Service Layer (CSL) and manages the processing of database access calls. IMS Connect then routes the response from ODBM back to the client.

IMS Universal drivers provide a multi-layer framework that provides additional programmatic control and access to IMS data if your application design or IT infrastructure calls for lower-level access. In addition to the standard CCI interface that is provided by the IMS Universal DB resource adapter, you can use the IMS Universal JDBC driver directly and issue SQL calls to access IMS data. When you want to make use of features in your hierarchical IMS databases that are not accessible through the relational-based JDBC driver, a low-level IMS database access API that provides DL/I capabilities, called the IMS Universal DL/I driver, is also available.

Regardless of which Universal Driver is chosen, it converts the commands into DRDA, which both IMS Connect and ODBM understand. For transactions that involve two-phase commit, IMS Connect builds the necessary Recovery Resource Services (RRS) structure to support the two-phase commit protocol. Because both IMS Connect and ODBM understand the DRDA protocol, they can fully support the data request flow from the client through TCP/IP to and from the IMS subsystem.

ODBM uses Structured Call Interface (SCI) services of the CSL for communication in the IMS Sysplex and uses MVS Program Call (PC) to communicate with IMS.

15.3.1 IMS Universal Drivers programming model

There are three main components that make it possible for IMS database to become an open architecture. In this section, we focus on the client-side libraries. The IMS Universal Drivers introduce three new open standard solutions to the IMS environment: Java EE Connector Architecture, JDBC, and DRDA. The support for DL/I will still exist.

The programming APIs support the following programming models:

- ▶ Common Client Interface (CCI) within a Java EE container
- ▶ JDBC
- ▶ Java implementation of DL/I
- ▶ Roll Your Own to allow direct use of the DRDA protocol

Overview of the IMS Universal drivers' programming approaches

The IMS Universal drivers offer multiple programming approaches, depending on your solution architecture and application needs.

IMS Universal DB resource adapter

You can use the standard Common Client Interface that the IMS Universal DB resource adapter provides to access IMS from a Java EE platform.

The capabilities of the Universal DB resource adapter are:

- ▶ Java EE Connector Architecture (JCA) 1.5 compliant
- ▶ Connection Management
 - Connection pooling
- ▶ Supports various types of interactions and programming models
 - JDBC
 - SQL
 - Java for DLI
- ▶ Security Management
 - RACF for authentication
 - ODBA does additional check for PSB (database) authorization
- ▶ Transaction Management
 - Global z/OS RRS transaction support
- ▶ Multi-platform support
 - Runs in WebSphere family of 31 and 64 bit application servers

IMS Universal JDBC driver

Using IMS support for JDBC you can write Java applications that can issue dynamic SQL calls to access IMS data and process the result set that is returned in tabular format. The IMS JDBC drivers support a subset of the SQL syntax with functionality that is limited to what the IMS database can do natively.

IMS Universal DL/I driver

This API provides support for all of the traditional DL/I capabilities and IMS functions by using the Java programming language. The IMS Universal DL/I driver can supplement your use of the IMS Universal JDBC driver or the IMS Universal DB resource adapter to issue database calls with more control over database characteristics that are inherently specific to IMS.

Which programming approaches to use to access IMS

There are several drivers for this solution depending on your IT infrastructure, solution architecture, and application design. You must choose the programming approach that is best for your development scenario. Table 15-1 provides potential scenarios and the ideal programming approach.

Table 15-1 Comparison of programming approaches

Scenario	Approach
Accessing IMS data through TCP/IP from a Java EE application server that resides on a distributed non-z/OS platform or a z/OS platform that is on a different logical partition (LPAR) from the IMS subsystem	Use the IMS Universal DB resource adapter
Accessing IMS data through TCP/IP from a Java application (outside of a Java EE application server) that resides on a distributed non-z/OS platform or a z/OS platform that is on a different LPAR from the IMS subsystem	Use the IMS Universal DL/I driver, IMS Universal JDBC driver, or the IMS Universal DB resource adapter in 'non-managed' mode
Accessing IMS data through TCP/IP from a non-Java application that resides on a distributed non-z/OS platform or a z/OS platform that is on a different LPAR from the IMS subsystem	Use a programming language of your choice to issue DRDA commands

In the next section, we discuss the Version 11 IMS Universal Drivers from an application development perspective. Using the industry standard APIs, the application is written to simply talk to the database, regardless of which Universal Driver is chosen.

15.3.2 Details on IMS Universal Drivers

In this section, we discuss the IMS Universal Drivers.

IMS Universal DB resource adapter

The back-end flow (IMS Connect, ODBM, IMS DB) shown in Figure 15-8 on page 332 is the same for each Universal Driver. For this reason, Figure 15-8 on page 332 only includes the IMS database to illustrate a more simplistic view of how the IMS Universal Driver is communicating with the IMS database.

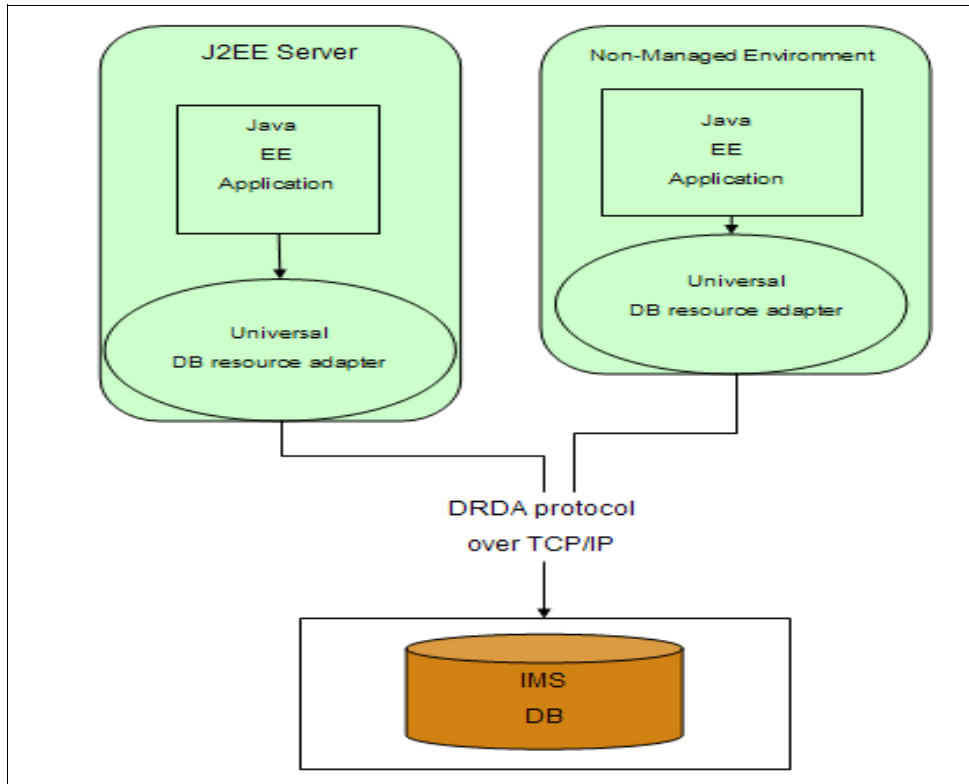


Figure 15-8 Topology of IMS Universal Drivers: DB resource adapter

The IMS Universal DB resource adapter is JCA 1.5 compliant. It can be deployed into a Java EE server or work in non-managed environment. It utilizes the CCI interface, which supports a subset of DL/I and SQL calls. In addition, there is support for the JDBC interface, which gives you the ability to write applications using SQL. When deploying the DB resource adapter in a Java EE environment, the Java application is a container that the Java EE Server manages or un-manages, if you chose to go with a standalone solution.

There are two versions of the IMS Universal DB resource adapter for optimized transaction management and performance. The two implementations of the DB resource adapter are:

- ▶ Local transaction support

The local DB resource adapter provides local transaction support when deployed on any supported Java EE application server. It is recommended to use one-phase commit.

- ▶ XA transaction support

This resource adapter provides both XA transaction and local transaction support when deployed on any supported Java EE application server connecting to IMS databases, it supports both two-phase commit and one-phase commit. Performance can be improved when using the local transaction DB resource adapter for one-phase commit.

IMS Universal JDBC driver

In Figure 15-9 on page 333 the Java EE application represents both the CCI Interface and the JDBC Interface.

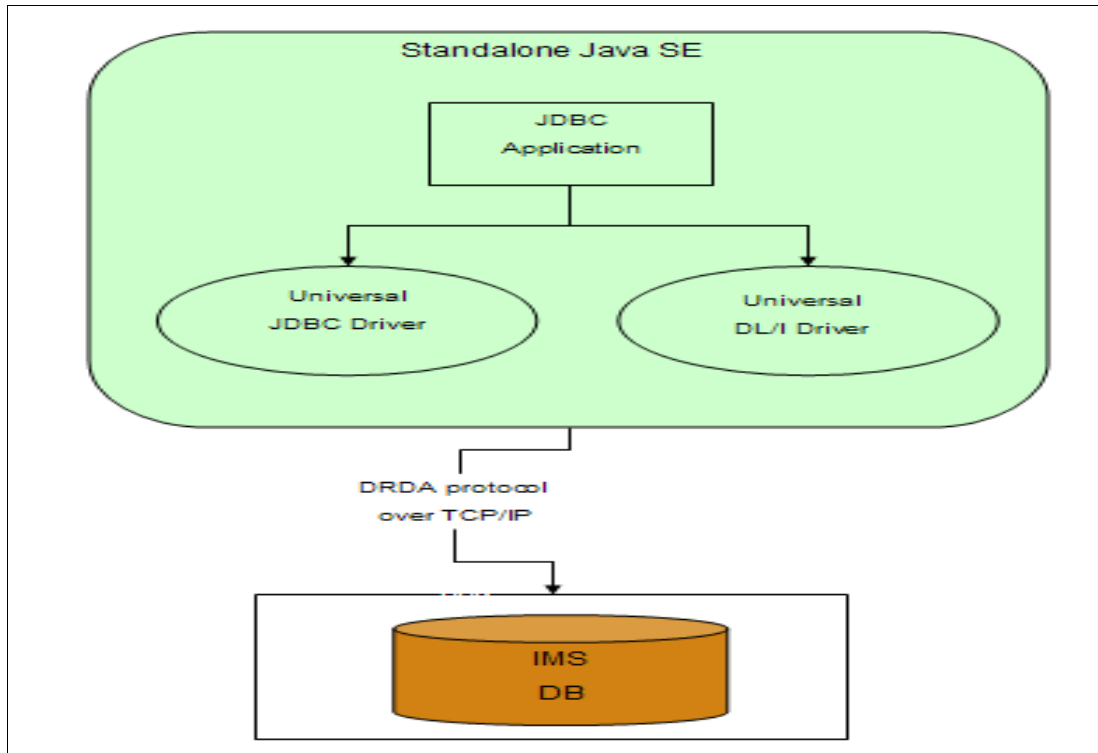


Figure 15-9 Topology of IMS Universal Drivers: JDBC driver and DLI Driver

The Universal JDBC driver is for SQL-based database connectivity to access IMS databases over TCP/IP.

JDBC is an application programming interface that Java applications use to access relational databases or tabular data sources. The JDBC API is the industry standard for database-independent connectivity between the Java programming language and any database that implemented the JDBC interface. The client uses the interface to query and update data in a database. Drivers are client-side adapters (they are installed in the client machine, not in the server) that convert requests from Java programs to a protocol that the database management system can understand.

IMS support for JDBC lets you write Java applications that can issue dynamic SQL calls to access IMS data and process the result set that is returned in tabular format. The IMS Universal JDBC driver is designed to support a subset of the SQL syntax with functionality that is limited to what the IMS database management system can process natively. The IMS Universal JDBC driver also provides aggregate function support, and ORDER BY, and GROUP BY support.

Situations can occur where a DLI call cannot be expressed using SQL (for example, lock classes do not translate to SQL). In these situations, use the Universal DLI driver. The good news is that you can use both the Universal JDBC and Universal DLI driver within the same application.

The sample Java application in Example 15-1 demonstrates the basic programming flow for a JDBC application using the IMS Universal JDBC driver.

Example 15-1 Sample standalone JDBC Application

```

import java.sql.Connection;
import java.sql.PreparedStatement;
  
```

```

import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import com.ibm.ims.dli.jdbc.IMSDataSource;
public class JDBCSTable {
    public static void main(String[] args) {
        IMSDataSource ds = new IMSDataSource();
        ds.setDatabaseName("MyDatabaseView");
        ds.setIMSSubsystemAlias("IMS1");
        ds.setDatabaseServer("ec0123.my.host.com");
        ds.setPortNumber(5555);
        ds.setDriverType(IMSDatasource.DRIVER_TYPE_4);
        ds.setUser("myUserId"); ds.setPassword("myPassword");
        Connection conn = null; try {
            conn = ds.getConnection();
            Statement st = conn.createStatement();

            //Find the last hospital in the database and get the HOSPCODE key
            ResultSet rs = st.executeQuery("SELECT HOSPCODE FROM PCB01.HOSPITAL");
            String lastHospCode = "";
            while(rs.next()){ lastHospCode = rs.getString("HOSPCODE").trim(); }
            //Find the first WARD and get the WARDNO key
            boolean resultSetAvailable = st.execute("SELECT WARDNO " +
                "FROM PCB01.WARD WHERE HOSPITAL_HOSPCODE='" + lastHospCode + "'");
            String firstWardNo = "";
            if(resultSetAvailable){
                rs = st.getResultSet();
                rs.next();
                firstWardNo = rs.getString("WARDNO").trim(); }
            else{
                System.out.println("No Wards are in the Hospital where HOSPCODE = " +
lastHospCode);
                System.exit(0); }

            //Now that we have keys for the WARD and HOSPITAL we can insert a new patient
            PreparedStatement pst = conn.prepareStatement("INSERT INTO PCB01.PATIENT " +
                "(HOSPITAL_HOSPCODE, WARD_WARDNO, PATNUM, PATNAME) VALUES(?,?,?,?)");
            pst.setString(1, lastHospCode);
            pst.setString(2, firstWardNo);
            pst.setString(3, "0222");
            pst.setString(4, "NEW PATIENT");

            int insertedRecords = pst.executeUpdate();
            System.out.println("Inserted " + insertedRecords + " Record(s)");

            int updatedRecords = st.executeUpdate("UPDATE PCB01.PATIENT " +
                "SET PATNAME='UPDATED NAME' WHERE PATNUM='0222'");
            System.out.println("Updated " + updatedRecords + " Record(s)");

            rs = st.executeQuery("SELECT * FROM PCB01.PATIENT WHERE PATNUM='0222'");
            rs.next();
            System.out.println("The PATNAME where PATNUM=0222 is: " +
                rs.getString("PATNAME").trim());
            int deletedRecords = st.executeUpdate("DELETE FROM PCB01.PATIENT " +
                "WHERE PATNAME='UPDATED NAME'");

```



```

System.out.println("Deleted " + deletedRecords + " Record(s)");

conn.commit();
conn.close();
} catch (SQLException e) {
    e.printStackTrace();
    try {
        if(conn!=null){
            conn.rollback();
            conn.close(); }
        } catch (SQLException e1) {
            e1.printStackTrace();
        }
    }
}
}
}

```

IMS Universal DL/I driver

If you notice both Figure 15-9 on page 333 and Figure 15-10 for JDBC and DL/I are similar, which is because no matter which API your application is written in, you can use the JDBC and DLI drivers interchangeably. It is possible to use the Java class library in both drivers. These drivers can be used in a one-phase and two-phase commit scenarios.

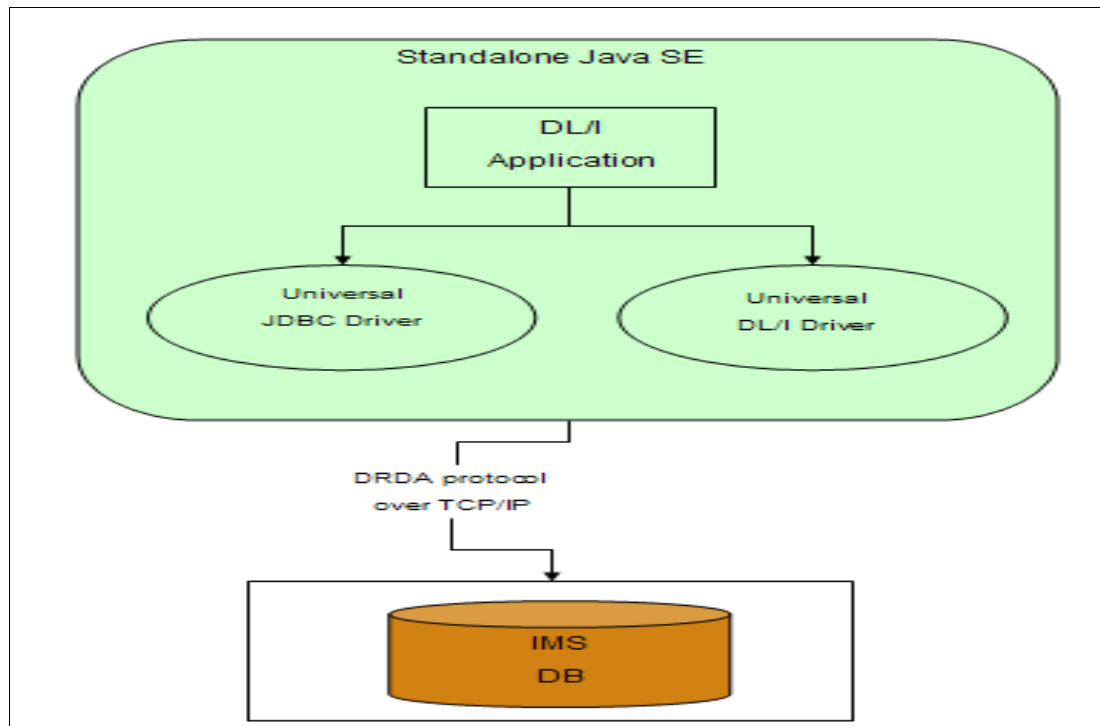


Figure 15-10 Topology of IMS Universal Drivers: JDBC driver and DL/I Driver

The Universal DL/I driver is used when you need to write granular queries to access IMS databases directly from a Java client in a standalone environment. Because of the fundamental differences between hierarchical databases and relational databases, sometimes the JDBC API does not provide access to the full set of IMS databases features. The IMS Universal DL/I driver is closely related to the traditional IMS DL/I database call

interface that is used with other programming languages for writing applications in IMS, and provides a lower-level access to IMS database functions than the JDBC API. By using the IMS Universal DLI driver, you can build segment search arguments (SSAs) and use the methods of the program control block (PCB) object to read, insert, update, delete, or perform batch operations on segments. You can gain full navigation control in the segment hierarchy.

The code fragment in Example 15-2 illustrates how to use the `batchRetrieve` method to retrieve the hospital name (HOSPNAME), ward name (WARDNAME), patient count (PATCOUNT), nurse count (NURCOUNT), and doctor count (DOCCOUNT) fields from the Hospital database.

Example 15-2 Sample standalone DLI Application

```
import com.ibm.ims.dli.*;
import com.ibm.ims.base.DLIException;

public class HospitalDLIReadClient {
    public static void main(String[] args) {
        PSB psb = null;
        PCB pcb = null;
        SSAList ssaList = null;
        Path path = null;
        PathSet pathSet = null;

        try {
            // establish a database connection
            IMSConnectionSpec connSpec = IMSConnectionSpecFactory.createIMSConnectionSpec();
            connSpec.setIMSSubsystemAlias("MyDBAlias");
            connSpec.setHostName("MyHostName");
            connSpec.setHostPortNumber(5555);
            connSpec.setDatabaseURL("MyDBURL");
            connSpec.setUserName("usr");
            connSpec.setPassword("password");
            connSpec.setDriverType(IMSConnectionSpec.DRIVER_TYPE_4);

            psb = PSBFactory.createPSB(connSpec);
            System.out.println("**** Created a connection to the IMS database");

            pcb = psb.getPCB("PCb01");
            System.out.println("**** Created PCB object");

            // specify the segment search arguments
            ssaList = pcb.getSSAList("HOSPITAL", "WARD");
            // add the initial qualification
            ssaList.addInitialQualification("HOSPITAL", "HOSPCODE",
                SSAList.GREATER_OR_EQUAL, 444);
            // specify the fields to retrieve
            ssaList.markFieldForRetrieval("HOSPITAL", "HOSPNAME", true);

            ssaList.markAllFieldsForRetrieval("WARD", true);
            ssaList.markFieldForRetrieval("WARD", "WARDNO", false);
            System.out.println("**** Created SSAList object");

            // issue the database call to perform a batch retrieve operation
            pathSet = pcb.batchRetrieve(ssaList);
            System.out.println("**** Batch Retrieve returned without exception");
```

```

System.out.println("**** Created PathSet object");

while(pathSet.hasNext()){
    path = pathSet.next();

    System.out.println("HOSPNAME: "+ path.getString("HOSPITAL", "HOSPNAME"));
    System.out.println("WARDNAME: "+ path.getString("WARD", "WARDNAME"));
    System.out.println("PATCOUNT: "+ path.getInt("WARD", "PATCOUNT"));
    System.out.println("NURCOUNT: "+ path.getInt("WARD", "NURCOUNT"));
    System.out.println("DOCCOUNT: "+ path.getShort("WARD", "DOCCOUNT"));

}
System.out.println("**** Fetched all rows from PathSet");
// close the database
connection psb.close();
System.out.println("**** Disconnected from IMS database");
} catch (DLIException e) {
    System.out.println(e); System.exit(0);
}
}
}

```

In summary, IMS Universal Drivers open the IMS architecture for remote access. With the flexibility of writing applications using the APIs that the IMS Universal Drivers provide, application developers can develop applications and deploy them in a Java EE server or a non-managed environment. Using the JDBC and DL/I API developers can write standalone Java applications. Because these APIs are based on industry standards, recent college graduates who are familiar with Java can write Java applications to talk to an IMS database without requiring deep mainframe skills.



Part 4

DataPower solutions for IMS

In part 4, we describe WebSphere DataPower SOA appliances, which are key elements in the IBM holistic approach to service-oriented architecture (SOA). These appliances are purpose-built, easy-to-deploy network devices to simplify, help secure, and accelerate your XML and Web services deployments.

There are three available devices to choose from:

- ▶ **WebSphere DataPower XML Accelerator XA35**
Capable of off loading overtaxed Web and application servers by processing XML, XSD, XPath and XSLT at wirespeed, the XA35 enables faster results from application investments.
- ▶ **WebSphere DataPower XML Security Gateway XS40**
Designed by some of the world's top XML and Web services security experts, the XS40 delivers a comprehensive set of configurable security and policy enforcement functions.
- ▶ **WebSphere DataPower Integration Appliance XI50**
IBM's hardware ESB, the XI50 is purpose-built for simplified deployment and hardened security, bridging multiple protocols and performing any-to-any conversions at wirespeed.

Chapter 16, "Using DataPower with IMS" on page 341 explains the functions and usefulness of these appliances and describes some usage scenarios with IMS.



Using DataPower with IMS

In this chapter, we discuss how the IBM WebSphere DataPower SOA appliances redefine the boundaries of middleware solutions by extending the SOA foundation with specialized, consumable, and dedicated SOA appliances that combine superior performance and hardened security for SOA implementations.

First we discuss the functions of the currently available 16.1, “IBM WebSphere DataPower XML Appliances” on page 342, and then we provide 16.2, “DataPower deployment scenarios and use cases” on page 344.

16.1 IBM WebSphere DataPower XML Appliances

There are three hardware appliances that you can use to simplify, help secure, and accelerate your XML and Web services deployments. We introduce all three in the order of increasing functionality.

16.1.1 WebSphere DataPower XML Accelerator XA35

As XML adoption within the enterprise increases, the growing number of slow-performing applications demonstrates that existing overtaxed software systems cannot support next-generation applications. Enterprises need a platform that addresses XML's performance, security, and management requirements head-on. Powered by unique purpose-built technology, IBM WebSphere DataPower XML Accelerator XA35 is a network device that can off-loading overtaxed servers by processing XML, XML Schema Definition, XML Path Language (XPath), and Extensible Stylesheet Language Transformations (XSLT) at wirespeed.

Features and benefits that are associated with the XA35 boosts the number of transactions per second while helping to lower the overall cost and complexity of portal applications:

- ▶ On demand publishing, wireless device support, and content syndication are all applications in which the XA35 provides transformation of XML into any format without requiring expensive and unwieldy infrastructure.
- ▶ Enable data and forms processing
Enterprises increasingly aggregate information from a number of sources, which includes telephone data entry, facsimile, online form submission, and manual data entry. Using XML and XSLT, enterprises can automate forms processing and normalize data received from any number of sources.
- ▶ Wirespeed performance
The purpose-built message processing engine delivers wirespeed performance for both XML to XML and XML to HTML transformations with increased throughput and decreased latency.
- ▶ Ease of use
The XA35 provides drop-in acceleration with virtually no changes to the network or application software.
- ▶ Reduced infrastructure costs
The XA35 fully parses, processes, and transforms XML with wirespeed performance and scalability to help reduce the need for stacks of servers.
- ▶ Lower development costs
The XA35 enables multiple applications to leverage a single, uniformed XML processing layer for all XML processing needs.
- ▶ Intelligent XML processing
The XA35 supports XML routing, XML pipeline processing, XML compression, XML/XSL caching, and other intelligent processing capabilities to help manage XML traffic.
- ▶ Advanced management
The XA35 provides real-time visibility into critical XML statistics, such as throughput, transaction counts, errors, and other processing statistics.

16.1.2 DataPower XML Security Gateway XS40

This appliance provides a security-enforcement point for XML and Web services transactions, including encryption, firewall filtering, digital signatures, schema validation, WS-Security, XML access control, XPath, and detailed logging. Some of its key features are:

- ▶ An XML/SOAP firewall

The DataPower XML Security Gateway XS40 filters traffic at wirespeed, based on information from layers 2 through 7 of the protocol stack, from field-level message content and SOAP envelopes to IP address, port/host name, payload size, or other metadata. Filters can be predefined with easy point-and-click XPath filtering GUI and automatically uploaded to change security policies based on the time of day or other triggers.

- ▶ XML/SOAP data validation

With its unique ability to perform XML Schema validation and message validation at wirespeed, the XS40 ensures that incoming and outgoing XML documents are legitimate and properly structured, which protects against threats, such as XML Denial of Service (XDoS) attacks, buffer overflows, or vulnerabilities that are created by deliberately or inadvertently malformed XML documents.

- ▶ Field level message security

The XS40 selectively shares information through encryption / decryption and signing / verification of entire messages or of individual XML fields. These granular and conditional security policies can be based on nearly any variable, which includes content, IP address, host name, or other user-defined filters

- ▶ Service virtualization

XML Web Services require companies to link partners to resources without leaking information about their location or configuration. With the combined power of URL rewriting, high-performance XSL transforms, and XML/SOAP routing, the XS40 can transparently map a rich set of services to protected back-end resources with high performance.

- ▶ Centralized policy management

Using the XS40's wirespeed performance enterprises can centralize security functions in a single, drop-in device that can enhance security and help to reduce ongoing maintenance costs. Simple firewall functionality can be configured through a GUI and running in minutes, and using the power of XSLT, the XS40 can also create sophisticated security and routing rules.

- ▶ Web Services management/service level management

The XS40 supports Web Services Distributed Management (WSDM), Universal Description Discovery and Integration (UDDI), Web Services Description Language (WSDL), and Dynamic Discovery, which helps to enable broad integration support for third-party management systems and unified dashboards in addition to robust support and enforcement for governance frameworks and policies.

Note: The DataPower XML Security Gateway XS40 includes all features from the DataPower XML Accelerator XA35.

16.1.3 DataPower Integration Appliance XI50

This appliance provides transport-independent transformations between binary, flat text files, and XML message formats. Visual tools are used to describe data formats and create mappings between different formats and to define message choreography. This appliance

can transform binary, flat-text, and other non-XML messages to help offer an innovative solution for security-rich XML enablement, ESBs, and mainframe connectivity.

The self-learning X150 provides drop-in acceleration with virtually no changes to the network or application software. No proprietary schemas, coding, or APIs are required to install or manage the device, and it supports popular XML Integrated Development Environments to help reduce the number of hours spent in developing and debugging XML applications.

For full product information about IBM WebSphere DataPower SOA Appliances, refer to:
<http://www.ibm.com/software/integration/datapower/index.html>

Note: The DataPower Integration Appliance X150 includes all features from the DataPower XML Accelerator XA35 and DataPower XML Security Gateway XS40.

16.2 DataPower deployment scenarios and use cases

DataPower SOA appliances provide a robust, secure platform for middleware integration that can be deployed in a wide array of deployment scenarios. In this section, we highlight the most common scenarios and use cases, but it is not intended to be an exhaustive list.

DataPower deployment scenarios

Figure 16-1 depicts common scenarios for deploying DataPower SOA appliances in the intranet, the demilitarized zone (DMZ), and a federated extranet (such as a business partner). Of particular importance is DataPower's capability to pass even the most stringent requirements for enterprise DMZ deployment.

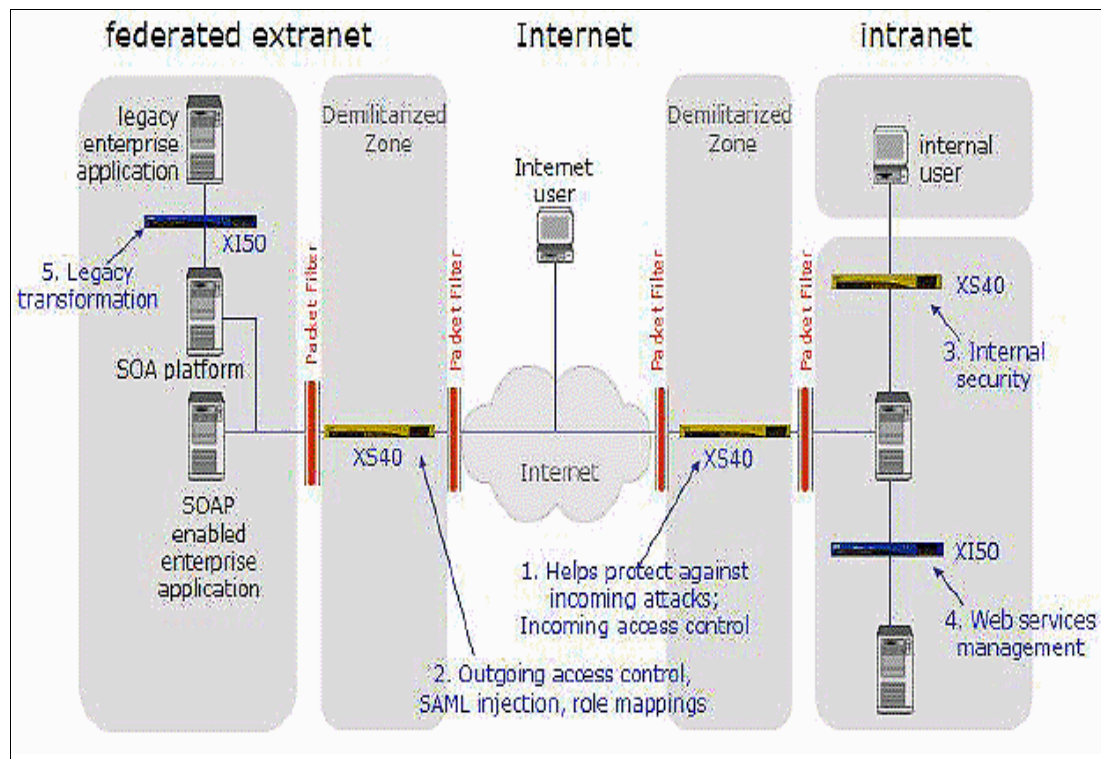


Figure 16-1 IBM DataPower deployment scenarios

DataPower use cases

We present use cases that are associated with monitoring and management, statistics visibility, Web Services management, service level management, XML threat protection and security, and functional acceleration.

Monitoring and management

DataPower has the ability for deep-content introspection at wirespeed, which enables the device to monitor service requests and responses without negatively affecting overall performance. DataPower appliances offer a number of different mechanisms for monitoring the traffic through the device from low-level service statistics to more elaborate service level management.

Statistics visibility

DataPower provides real-time visibility into critical statistics, such as throughput, transaction counts, errors, message size, and other processing statistics. Data network level analysis is provided and includes server health information, traffic statistics, management, and configuration data.

Web Services management

DataPower supports Web Services Distributed Management, Universal Description, Discovery, and Integration, Web Services Description Language, and Dynamic Discovery.

Service Level management

DataPower allows the specification of Quality of Service (QoS) policies that shape or throttle traffic based on service-level criteria, which enables the prioritization of service requests in support of business goals.

DataPower integrates with various monitoring products, such as IBM Tivoli Enterprise Monitoring and Netegrity SiteMinder. Figure 16-2 illustrates how DataPower appliances can forward service-level information to the IBM Tivoli Composite Application Management for SOA, which in turn presents graphical views of service performance in the SOA.

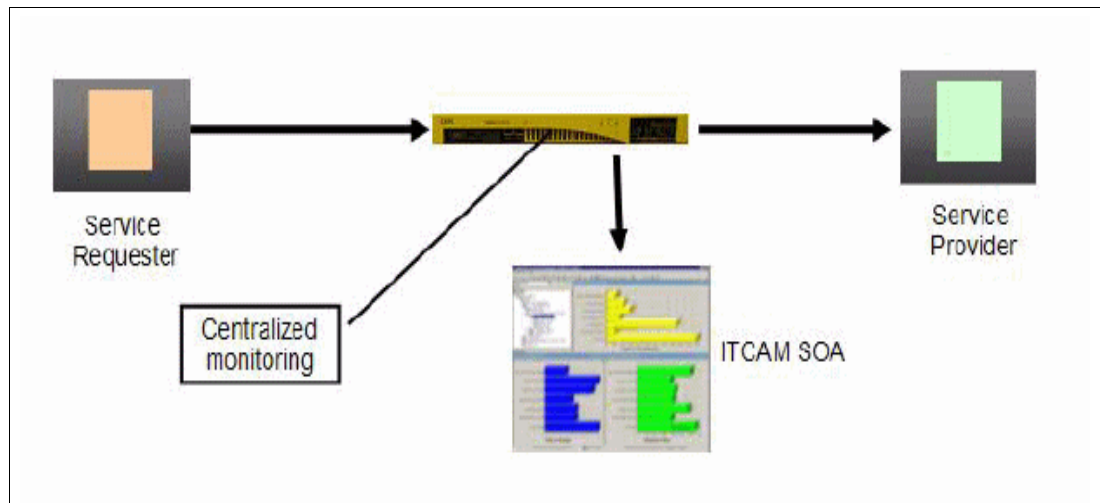


Figure 16-2 Centralized monitoring and management with DataPower

XML threat protection and security

Traditional firewalls only protect traffic at the Internet Protocol (IP) level. Web services effectively tunnel through the IP firewall layer through standard HTTP(s) and expose the organization's applications to completely new threats. Stated simply, we need to ensure that

only valid requests for valid services from genuine clients penetrate the enterprise boundary. An “XML firewall” is needed.

Do not underestimate the seriousness of these new threats. The following IBM DeveloperWorks article clearly defines the breadth and seriousness of different attacks that are possible to any service that is exposed using XML.

http://www.ibm.com/developerworks/websphere/techjournal/0603_col_hines/0603_col_hines.html

The article concludes with the comments:

“To truly harden a system using Web services, several important security steps (recommended by Gartner and others) are required, including:

- ▶ Inspect messages for well-formation
- ▶ Validate schema
- ▶ Verify digital signatures
- ▶ Sign messages
- ▶ Implement service virtualization to mask internal resources through XML transformation and routing
- ▶ Encrypt data at the field level”

Systems hosting Web services, particularly public Internet-facing services, must seriously consider the case for hardened gateway devices acting as XML firewalls to protect systems from XML threats.”

DataPower appliances address these issues by delivering a robust XML firewall for the enterprise.

Functional acceleration

Customers need XML for implementing SOA, but cannot afford to spend precious CPU cycles processing it. There is an evolution towards using dedicated hardware for performing repetitive XML tasks, such as parsing, schema validation, and XML Stylesheet Language (XSL) translation. Service protocols that are based on XML also lack any inherent built-in security mechanisms. SOAP over HTTP passes potentially sensitive data in plain text over the network. Although there are emerging standards, such as WS-Security, to help deal with security concerns, implementing these standards further drains computing resources on critical servers.

The IBM line of DataPower SOA appliances helps to address the performance and security needs of enterprise-level SOA architectures by off-loading the XML processing onto dedicated hardware, which frees the CPU resources of application servers and middleware platforms to provide higher service throughput. The performance advantage of DataPower appliances are often close to seventy times higher than when using general purpose systems alone.

16.2.1 Protocol and format bridging

With DataPower, services can be exposed using different formats and protocols to the ones in which they are implemented. Ultimately, this translation capability encourages reuse and lowers total cost of ownership (TCO) by avoiding costly re-implementation of existing services. More interestingly, DataPower appliances can expose services using several different formats and protocols at once, thus supporting a wide range of clients.

Protocols

Services can be exposed and called using any combination of the typical protocols used for passing SOAP and XML messages in an SOA, such as HTTP, HTTPS, and Java Message Service (JMS). Direct communication with WebSphere MQ and IMS Connect is also supported.

Any-to-any Transformation Engine

If the enterprise's standard protocols reach beyond the commonly accepted Web Services data formats, appliances can parse and transform the following data formats:

- ▶ Binary
- ▶ Flat text and XML messages, including EDI, COBOL Copybook, ISO 8583 (International Organization for Standardization 8583 standard for financial transaction card originated messages)
- ▶ ASN.1 (Abstract Syntax Notation One)
- ▶ ebXML (Electronic Business using eXtensible Markup Language)

Unlike approaches that are based on custom programming, DataPower's patented DataGlue technology uses a fully declarative, metadata-based approach that delivers wirespeed performance.

Figure 16-3 shows an example where an existing enterprise service, such as an IMS application, can leverage DataPower to provide its Web service facade. In this example, DataPower is converting the SOAP format over the HTTP transport to the Cobol Copybook format over the MQ transport. DataPower's support for transports and protocols is orthogonal, which means it can support any format over any transport to any other combination, for example, Figure 16-3 might just as easily be a Cobol/MQ facade to an existing SOAP/HTTP Web service.

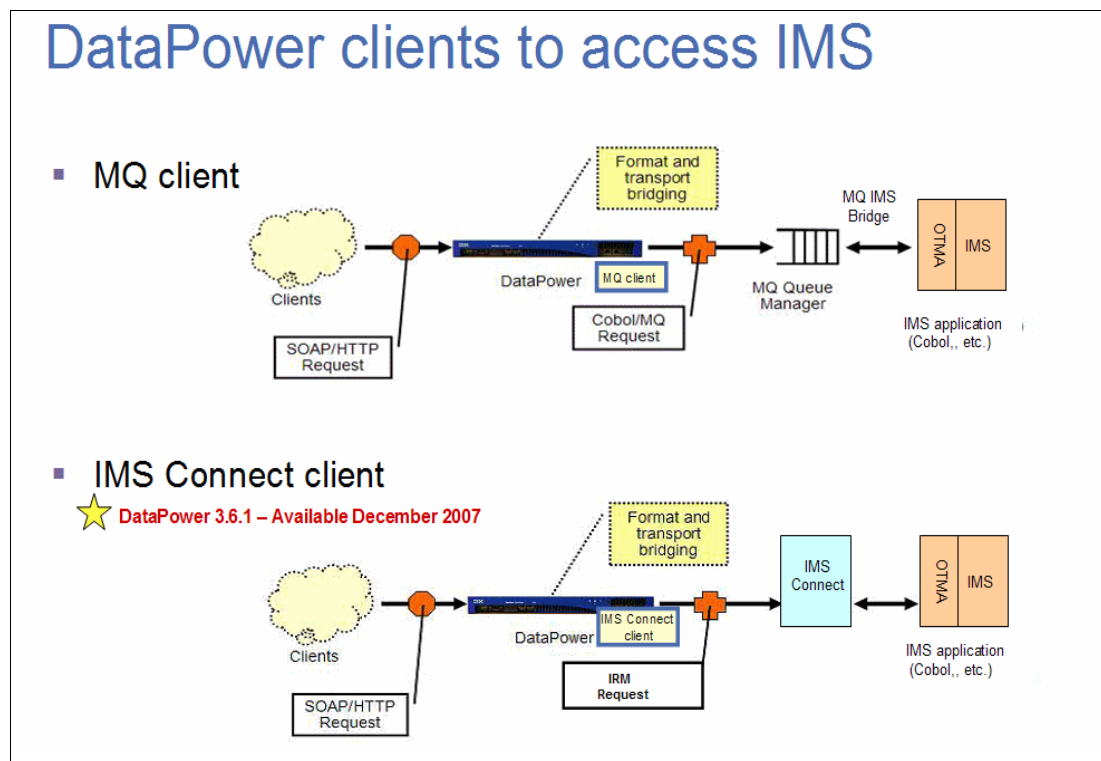


Figure 16-3 DataPower clients access to IMS

In DataPower firmware V3.6.1, a feature called "IMS Protocol Support" adds support to allow Multi-Protocol Gateway services to accept IMS connections from clients and connect to IMS-based applications.

Some high-level details on DataPower protocols are:

- ▶ DataPower can interface with IMS Connect since firmware Version 3.6.1
- ▶ It provides synclevel none, commit mode 1 support only
- ▶ There are two main IMS protocol support scenarios:
 - Act as IMS Connect proxy to IMS Connect clients: Main use case here is for existing IMS Connect clients who want to make in flight modifications to header and payload without changing client code or IMS.
 - Provide Web service facade to IMS Connect transactions: Use case here is to make use of the strong Web service features in DataPower to quickly enable Web service support for IMS Connect.

16.2.2 Configuring and using DataPower

DataPower provides a powerful Web Graphical User Interface (Web GUI), as shown in Figure 16-4. There is a palette of common mediations (actions) that can be dropped into the message processing policy. In addition to the Web GUI, DataPower provides a command line interface (CLI) that is accessible through SSH (Secure Shell is a network protocol that allows data to be exchanged using a secure channel between two networked devices) and Telnet (TELEcommunication NETwork). An Eclipse plug-in enables tooling support for configuration. Multiple appliances can be managed together as part of a set through the use of IBM Tivoli Composite Appliance Management System Edition for WebSphere DataPower (ITCAMSE for WDP).

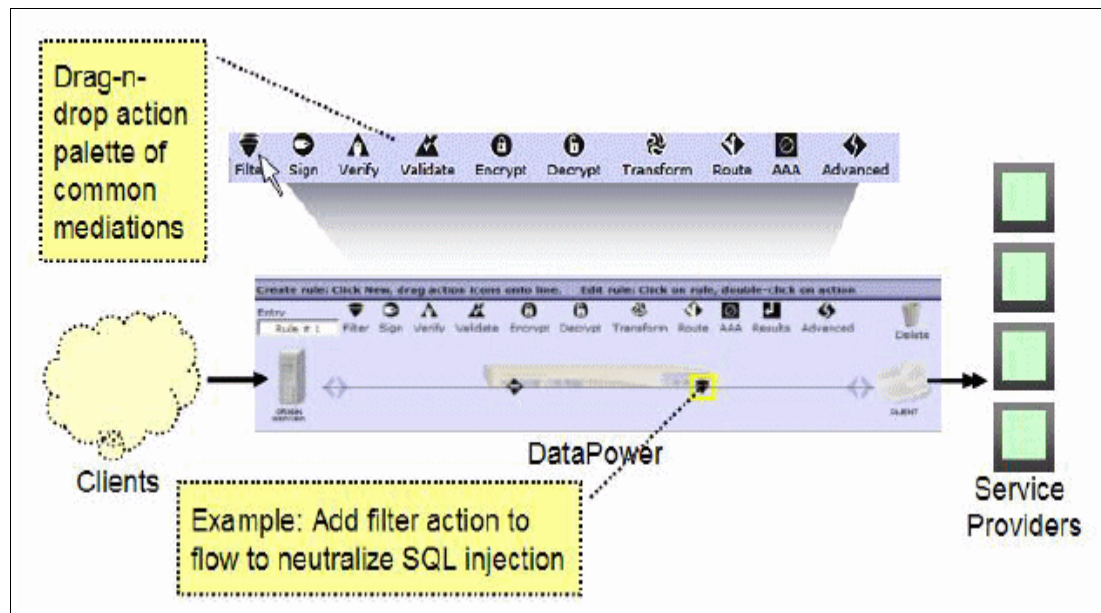


Figure 16-4 DataPower Web GUI

As soon as the DataPower appliance is connected to the network you can connect into it by pointing your browser to the Network IP address of the box. After you login, you see the control panel, as presented in Figure 16-5 on page 349.



Figure 16-5 DataPower XI50 control panel

Figure 16-6 on page 350 displays the main menu where you configure the Host, Port, Conversion, and a Client _ID prefix information for IMS Connect:

- ▶ Host
 - Specify the host name or IP address of the IMS Connect server.
- ▶ Port
 - Specify the port on which the IMS Connect server is running.
- ▶ EBCDIC header conversion
 - This option can be turned on for converting the headers to EBCDIC. The IMS Connect exit should be able to process EBCDIC data. Some IMS Connect exits can handle both UTF-8 and EBCDIC. This conversion affects only the headers. Use transformation to do any data conversion in the policy.
- ▶ Generate client ID prefix
 - A two-letter prefix for the generated client ID. "DP" is used if not specified.

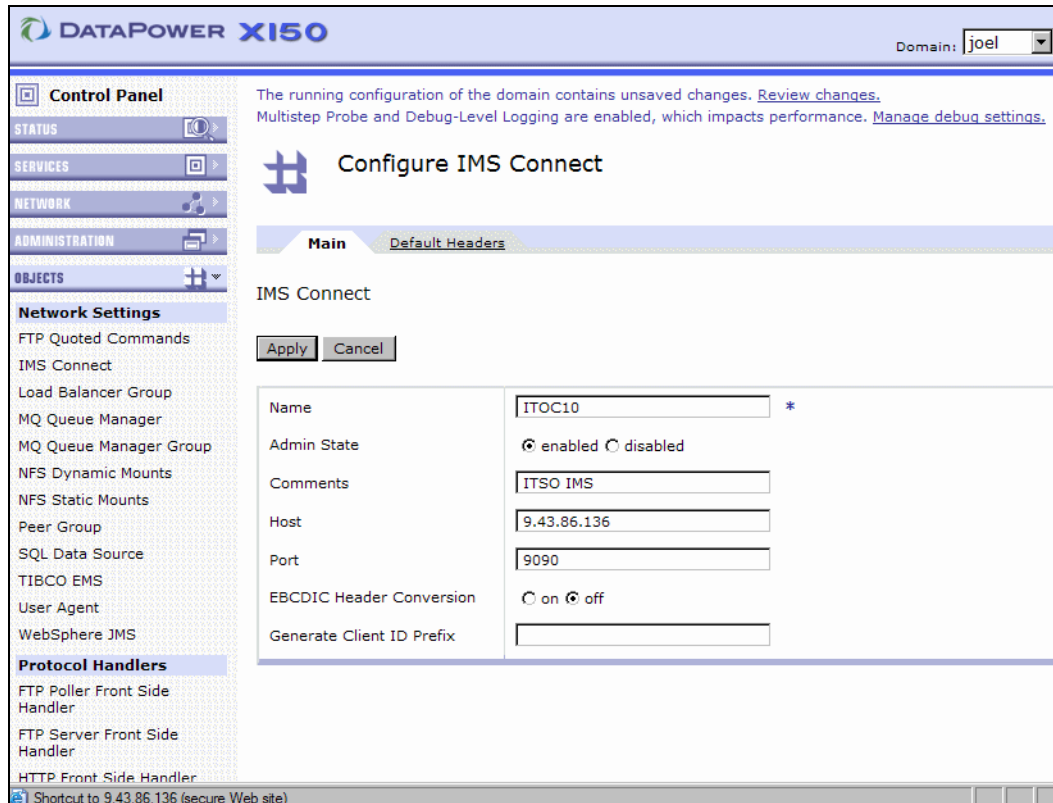
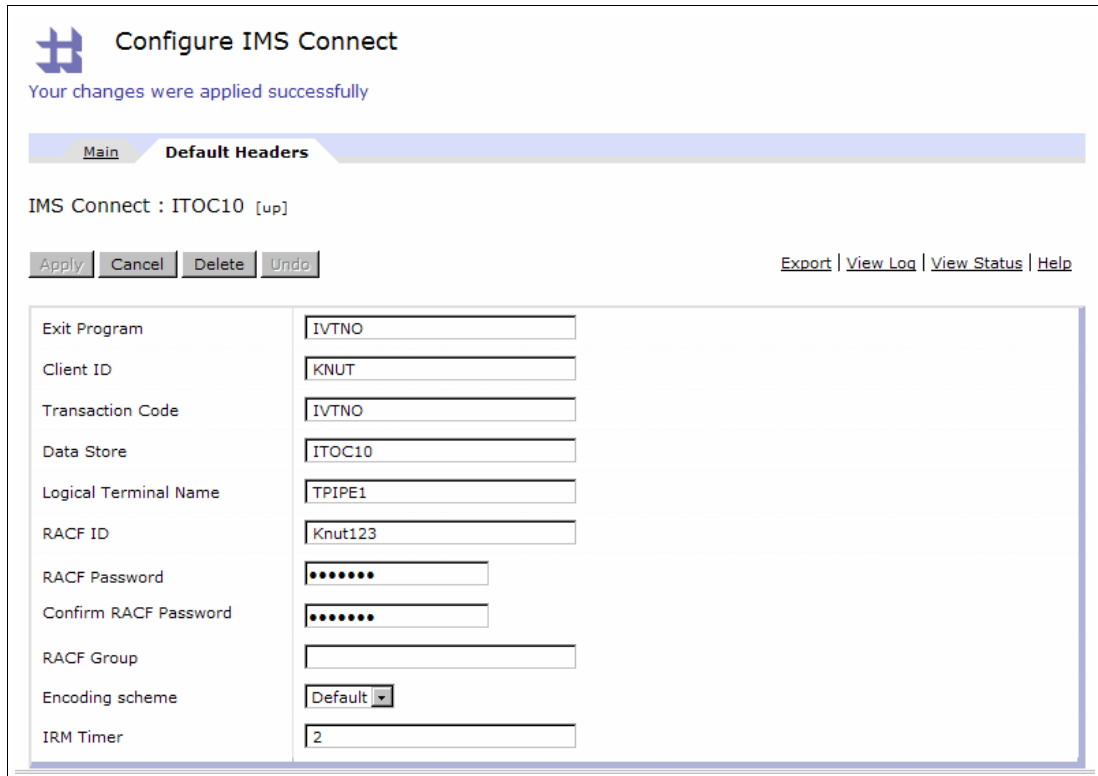


Figure 16-6 Configure IMS Connect access through DataPower

Figure 16-7 on page 351 displays the default header menu where you specify:

- ▶ Exit Program: The exit program to use for all the IMS connections.
- ▶ Client ID: A string of 1 to 8 uppercase alphanumeric (A through Z, 0 to 9) or special (@, #, \$) characters, left justified, and padded with blanks. It specifies the name of the client ID that is used by IMS Connect. If this string is not supplied from the client, then the user exit generates it.
- ▶ Transaction code: The IMS transaction code to invoke.
- ▶ Data store: It specifies the Datastore name (IMS destination ID).
- ▶ Logical terminal name: The LTERM override value to be used by OTMA.
- ▶ RACF ID: The plain text string sent to the server for identifying the client.
- ▶ RACF Password: The host security password used to login to the IMS Connect server. Enter the password twice to confirm its accuracy.
- ▶ RACF Group: The group the Host security ID belongs to.
- ▶ Encoding scheme: Select the Unicode encoding schema. Leave as (none) to be set dynamically in the IMS header.
- ▶ IRM Timer: Set the IRM_TIMER: Set the value to an appropriate wait time for IMS to return data to IMS Connect. See IMS Connect documentation for details. An example value of 21 would set an IRM Timer value of 0.21 sec.



Configure IMS Connect

Your changes were applied successfully

Main **Default Headers**

IMS Connect : ITOC10 [up]

Apply Cancel Delete Undo

[Export](#) | [View Log](#) | [View Status](#) | [Help](#)

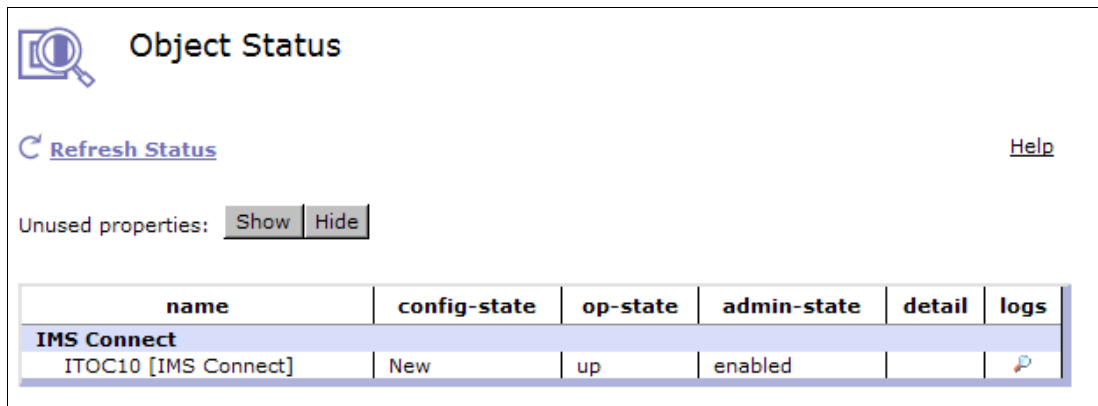
Exit Program	IVTNO
Client ID	KNUT
Transaction Code	IVTNO
Data Store	ITOC10
Logical Terminal Name	TPIPE1
RACF ID	Knut123
RACF Password	*****
Confirm RACF Password	*****
RACF Group	
Encoding scheme	Default
IRM Timer	2

Figure 16-7 DataPower configure IMS connect default headers

On the View Status Panel, you see the Object Status of the recently added IMS Connect Object. Figure 16-8 presents the configuration object operational state, which can be one of the following:

Status:

- ▶ Invalid: Invalid Configuration
- ▶ Saved: Persisted Configuration
- ▶ New: New Configuration
- ▶ Modified: Modified Configuration
- ▶ Deleted: Deleted Configuration
- ▶ External: External Configuration



Object Status

[Refresh Status](#) [Help](#)

Unused properties: [Show](#) [Hide](#)

name	config-state	op-state	admin-state	detail	logs
IMS Connect					
ITOC10 [IMS Connect]	New	up	enabled		

Figure 16-8 Object Status

Figure 16-9 displays the system log for IMS Connect where the activity performed for that connector is presented. In this example, the msgid of 'Event Code 0x00360013 - Configured', indicates a valid configuration. The object is configured but not active at this time

The screenshot shows a web-based interface for viewing system logs. At the top, there is a magnifying glass icon and the title "System Log for IMS Connect 'ITOC10'". Below the title, there is a "Refresh Log" button and three dropdown menus for "Target" (set to "default-log"), "Filter" (set to "(none)"), and another "Filter" (set to "(none)"). The current time is displayed as "14:28:07 on 2008-08-21". Below this is a table with columns: time, category, level, tid, dir, client, msgid, and message. A "Show last 50 100 all" link is on the right. The table shows a log entry for "Thu Aug 21 2008" with the following details: time: 14:08:55, category: mgmt, level: info, tid: 53167, msgid: 0x00360013, and message: ims (ITOC10): Configured.

time	category	level	tid	dir	client	msgid	message
Thu Aug 21 2008							
14:08:55	mgmt	info	53167			0x00360013	ims (ITOC10): Configured.

Figure 16-9 System log for IMS Connect

In summary, we believe that examining the use of DataPower appliances while you are planning for IMS SOA implementations is worth the effort.

Part 5



Appendixes

Appendix A contains code snippets that are related to items that we described in earlier chapters.



Sample code snippets

This appendix contains code snippets that are related to items that we described in earlier chapters:

- ▶ “Asynchronous callout to a SLSB” on page 236 references code snippet A.1, “Asynchronous callout to a Stateless Session Bean” on page 355
- ▶ “Asynchronous callout to an MDB” on page 237 references code snippet A.2, “Asynchronous callout to a Message Driven Bean” on page 357
- ▶ “Synchronous callout to an SLSB” on page 241 references code snippet A.3, “Synchronous callout to a Stateless Session Bean” on page 360
- ▶ “Synchronous callout to an MDB” on page 242 references code snippet A.4, “Synchronous callout to a Message Driven Bean” on page 361
- ▶ “Creating a feed from an IMS application in InfoSphere MashupHub” on page 291 references code snippet A.5 “Feed from an IMS application in MashupHub” on page 362
- ▶ “Web Services for DLI databases (IMS V11 and IMS V10 with SPE)” on page 306 references code snippet A.6, “WSDL files for a DLIModel generated Web Services Service” on page 364
- ▶ “Web Services for DLI databases (IMS V11 and IMS V10 with SPE)” on page 306 references code snippet A.7, “XSD files for a DLIModel generated Web Services Service” on page 366

A.1 Asynchronous callout to a Stateless Session Bean

The code in A-1 is “ASIS”. Some details can be changed at the availability date.

Example: A-1 Code snippet for asynchronous to SLSB in WebSphere Application Server

```
package ejbs;
import javax.naming.InitialContext;
import javax.resource.cci.Connection;
import javax.resource.cci.ConnectionFactory;
import javax.resource.cci.Interaction;
import javax.resource.cci.Record;
```

```

import com.ibm.connector2.ims.ico.IMSConnectionSpec;
import com.ibm.connector2.ims.ico.IMSInteractionSpec;
/**
 * Bean implementation class for Enterprise Bean: SLSB4IMSASyncCallOut1
 */
public class SLSB4IMSASyncCallOut1Bean implements javax.ejb.SessionBean {
    /**
     * deleted lines without interest for IMS callout
     */
    static final long serialVersionUID = 3206093459760846163L;
    public void start() { //_____ startup EJB called method
        try {
            InitialContext initCtx = new InitialContext(); // (1)
            // JNDI lookup returns connFactory
            ConnectionFactory connFactory = (ConnectionFactory) initCtx
                .lookup("java:comp/env/ibm/ims/IMSTarget"); // -> ConnectionFactory (2)
            IMSConnectionSpec connSpec = new IMSConnectionSpec(); // (3)
            Connection connection = connFactory.getConnection(connSpec); // (4)
            Interaction interaction = connection.createInteraction(); // (5)
            IMSInteractionSpec interactionSpec = new IMSInteractionSpec(); // (6)
            // set the commit mode and sync level
            interactionSpec.setCommitMode(0); // (7)
            interactionSpec.setSyncLevel(1); // (7)
            // An 8-character queue name that the asynchronous messages to be retrieved from
            String calloutQueueName = new String("CALLOUTQ");
            // Set the asynchronous queue name for the callout message
            interactionSpec.setAltClientID(calloutQueueName); // (8)
            // Set interactionVerb to retrieve async output with a timeout
            interactionSpec
                .setInteractionVerb(IMSInteractionSpec.SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT); //(9)
            interactionSpec.setExecutionTimeout(3600000);
            Record calloutMsg = null; // (9) should be a DataBean
            // looping for message from IMS
            for (;;) {
                try {
                    // Execute the interaction
                    interaction.execute(interactionSpec, null, calloutMsg); // (10)
                    // process received message
                    // .....
                    // ----- if response required -----
                    // Use again a CCI interaction for sending back the response: SEND_ONLY
                    // Response should contain IMSTRAN, it is a new transaction
                    //-----
                    // build new interaction if required to send response IMSTRAN // (11)
                    //-----}
                } catch (Exception e2) {
                    // if the exception is an execution timeout error,
                    // you can either do nothing and continue to loop
                    // or process the error and then break the loop
                    break;
                }
            }
            interaction.close();
            connection.close();
        } catch (Exception e1) {
        }
    }
}

```

1. Set initial context.
2. Use context to lookup the Connection Factory with JNDIName
"java:comp/env/ibm/ims/IMSTarget"
3. Instantiate connectionSpec.
4. Get connection from connectionFactory, augmented with connectionSpec.
5. Create interaction from connection.
6. Instantiate interactionSpec.
7. Set synclevel, commit as indicated in interactionSpec.
8. Set destination for reading callout message in interactionSpec.
9. Set interactionVerb.
10. Message must be received in a databean (javax.resource.cci.Record).
11. Execute interaction with IMS to retrieve callout message.
12. If required, build response message, use existing connection to send new transaction to IMS as "send_only" interaction.

A.2 Asynchronous callout to a Message Driven Bean

The code in A-2 is "ASIS". Some details can be changed at the availability date.

Example: A-2 Code snippet for asynchronous callout to MDB in WeBSphere Application Server

```

package ejbs;
import javax.resource.cci.Record;
/**
 * Bean implementation class for Enterprise Bean: MDB4IMSASyncCallOut1
 */
public class MDB4IMSASyncCallOut1Bean
    implements
        javax.ejb.MessageDrivenBean,
        commonj.connector.runtime.InboundListener { // (1)
    private javax.ejb.MessageDrivenContext fMessageDrivenCtx;
    /**
     * deleted lines without interest for IMS callout
     */
    /**
     * onMessage NOT used for asynchronous call
     */
    public javax.resource.cci.Record onMessage(javax.resource.cci.Record arg0){
        return null;
    }
    /**
     * onNotification
     */
    public void onNotification(javax.resource.cci.Record arg0)
        throws javax.resource.ResourceException { // (2)
        processMessageReceived(arg0); // (3)
        // ----- if response required -----
        // Use again a CCI interaction for sending back the response: SEND_ONLY
        // Response should contain IMSTRAN, it is a new transaction (4)
        //-----
        // build new interaction if required to send response IMSTRAN
        //-----}

```

```

}
/**
 * processMessageReceived
 */
public void processMessageReceived(Object event){
    // examine the message and process accordingly
}
}
}

```

1. MDB must implement this interface.
2. Message of type (javax.resource.cci.Record) received in onNotification().
3. Process the received message.
4. If required, build response message, setup connection to send new transaction to IMS as “send_only” interaction.

The MDB code is much more simple than the SLSB code because a listener is used. This listener must know where to listen. This information is declared in the *ActivationSpec*, as shown in Figure A-1. The activationSpec is one of the elements of the MDB.

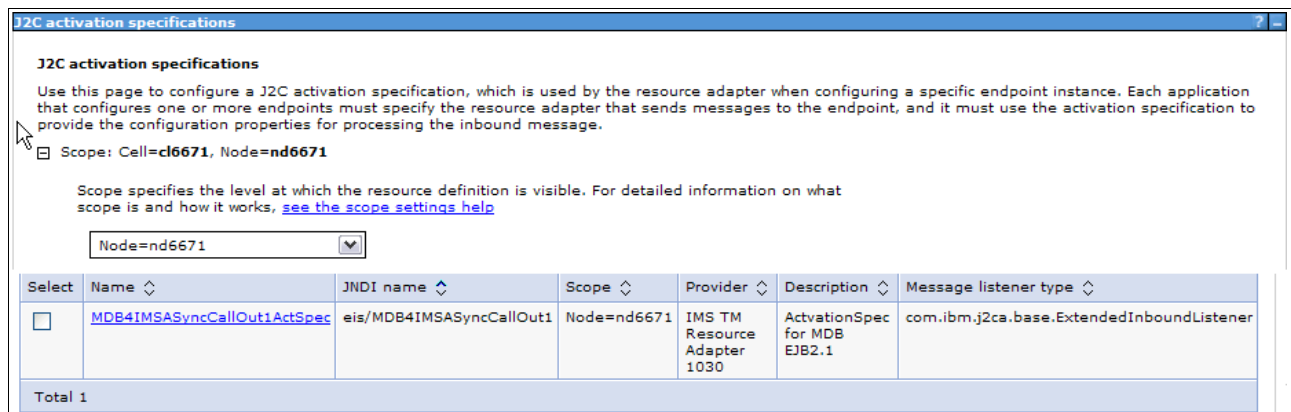


Figure A-1 ActivationSpec

For each MDB, an activationSpec must be defined by the administration of WebSphere Application Server. It is declared under the TM Resource Adapter provider. See Figure A-2 on page 359.

J2C activation specifications > **MDB4IMASyncCallOut1ActSpec** > **Custom properties**

Use this page to specify custom properties that your enterprise information system (EIS) requires for the resource providers and resource factories that you configure. For example, most database vendors require additional custom properties for data sources that access the database.

⊞ Preferences

Name	Value	Description	Required
queueNames			true
portNumber	9999		true
hostName	wtsc67.itso.ibm.com		true
dataStoreName	IMSL		true
SSLKeyStoreName			false
retryInterval	60000		false
SSLKeyStorePassword			false
filterFutureEvents	false		false
pollPeriod	2000		false
pollQuantity	10		false
BONamespace			false
deliveryType	ORDERED		false
SSLEnabled	false		false
assuredOnceDelivery	true		false
stopPollingOnError	false		false
eventTypeFilter			false
userName			false
retryLimit	0		false
SSLTrustStorePassword			false
SSLTrustStoreName			false

Page: 1 of 2 Total 25

Figure A-2 ActivationSpec properties

In Figure A-3, you find the properties that must be specified for the activationSpec of the MDB. Mainly they correspond to IPAddr, portnr of IMS Connect, and the IMS datastore name.

The link between activationSpec and MDB is through the EJB(MDB) deployment descriptor, which is part of the deployed Java EE application together with the MDB itself.

WebSphere Bindings
The following are binding properties for the WebSphere Application Server.

Listener Port
Listener port name:

JCA Adapter
ActivationSpec JNDI name:
ActivationSpec Authorization Alias:
Destination JNDI name:

Figure A-3 activationSpec name in Deployment Descriptor of MDB

A.3 Synchronous callout to a Stateless Session Bean

The code in A-3 is “ASIS”. Some details can be changed at the availability date.

Example: A-3 Code snippet for synchronous callout to SLDB in WeBSphere Application Server

```
package ejbs;
import javax.naming.InitialContext;
import javax.resource.cci.Connection;
import javax.resource.cci.ConnectionFactory;
import javax.resource.cci.Interaction;
import javax.resource.cci.Record;
import com.ibm.connector2.ims.ico.IMSConnectionSpec;
import com.ibm.connector2.ims.ico.IMSInteractionSpec;
/**
 * Bean implementation class for Enterprise Bean: SLSB4IMSSyncCallOut1
 */
public class SLSB4IMSSyncCallOut1Bean implements javax.ejb.SessionBean {

    static final long serialVersionUID = 3206093459760846163L;
    private javax.ejb.SessionContext mySessionCtx;
    /**
     * deleted lines without interest for IMS callout
     */
    public void start() {
        try {
            InitialContext initCtx = new InitialContext(); // (1)
            // JNDI lookup returns connFactory
            ConnectionFactory connFactory = (ConnectionFactory) initCtx
                .lookup("java:comp/env/ibm/ims/IMSTarget"); // (2)
            IMSConnectionSpec connSpec = new IMSConnectionSpec(); // (3)
            Connection connection = connFactory.getConnection(connSpec); // (4)
            Interaction interaction = connection.createInteraction(); // (5)
            IMSInteractionSpec interactionSpec = new IMSInteractionSpec(); // (6)
            // set the commit mode and sync level
            interactionSpec.setCommitMode(0); // (7)
            interactionSpec.setSyncLevel(1); // (7)
            // A 8 character name that the asynchronous messages to be retrieved from
            String calloutQueueName = new String("CALLOUTQ");
            // Set the queue name for the callout message
            interactionSpec.setAltClientID(calloutQueueName); // (8)
            // Set InteractionVerb for retrieve async output with a very large // timeout
            interactionSpec
                .setInteractionVerb(IMSInteractionSpec.SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT); // (9)
            interactionSpec.setExecutionTimeout(360000);
            // Execute the interaction
            Record calloutMsg = null; /* Temporary should be databean */ // (10)
            // looping for message from IMS
            for (;;) {
                try {
                    interaction.execute(interactionSpec, null, calloutMsg); // (11)
                    // Get Correlation token
                    // ----> following is for synchronous callout
                    String corrToken = interactionSpec.getSyncCalloutToken(); // (12)
                    // Further processing on the calloutMsg
                    // Send back the response
                    interactionSpec.setInteractionVerb
                        (com.ibm.connector2.ims.ico.IMSInteractionSpec.SYNC_SEND); // (13)
                    // ----> following is correlating callout and response, if new InteractionSpec
                    interactionSpec.setSyncCalloutToken(corrToken); // (14)
                }
            }
        }
    }
}
```

```

    // Execute the interaction
    Record responseMsg = null; /* Temporary */
    interaction.execute(interactionSpec, responseMsg, null); // (15)
    interaction.close();
    connection.close();
} catch (Exception e1) {
}
}
}

```

1. Set initial context.
2. Use context to lookup the Connection Factory with JNDIName **"java:comp/env/ibm/ims/IMSTarget"**
3. Instantiate connectionSpec.
4. Get connection from connectionFactory.
5. Create interaction from connection.
6. Instantiate interactionSpec.
7. Set synclevel, commit as indicated in interactionSpec.
8. Set destination for reading callout message in interactionSpec.
9. Set interactionVerb.
10. Message must be received in a databean (javax.resource.cci.Record).
11. Execute interaction with IMS to retrieve callout message.
12. Recuperate correlation token.
13. Set interactionverb for sending response.
14. Set correlation token.
15. Build response message of type databean (javax.resource.cci.Record), and use existing connection to send new transaction to IMS as "send_only" interaction.

A.4 Synchronous callout to a Message Driven Bean

The code in A-4 is "ASIS". Some details can be changed at the availability date.

Example: A-4 Code snippet for synchronous callout to MDB in WeBSphere Application Server

```

package ejbs;
import javax.resource.cci.Record;
/**
 * Bean implementation class for Enterprise Bean: MDB4IMSSyncCallOut1
 */
public class MDB4IMSSyncCallOut1Bean
    implements
        javax.ejb.MessageDrivenBean,
        commonj.connector.runtime.InboundListener { // (1)
    private static final long serialVersionUID = 1L;
    private javax.ejb.MessageDrivenContext fMessageDrivenCtx;
    /**
     * deleted lines without interest for IMS callout
     */
    /**
     * onMessage

```

```

*/
public javax.resource.cci.Record onMessage(javax.resource.cci.Record arg0){ // (2)
    processMessageReceived(arg0); // (3)
    Record testRecord = null; /* temporarily */ // (4)
    //INPUTMSG testRecord = new INPUTMSG();
    //testRecord.setIn_ll((short)74);
    //testRecord.setIn_zz((short)0);
    //testRecord.setIn_data(...);
    return testRecord; // (5)
}
/**
 * onNotification NOT used for synchronous call
 */
public void onNotification(javax.resource.cci.Record arg0)
    throws javax.resource.ResourceException {
}
/**
 * processMessageReceived
 */
public void processMessageReceived(Object event){
    // examine the message and process accordingly
}
}

```

1. MDB must implement this interface.
2. Message of type `javax.resource.cci.Record` received in `onMessage()`.
3. Process the received message.
4. Prepare response record of type `databean (javax.resource.cci.Record)`.
5. Return response (automatically correlated with request).

The MDB code is much more simple than the SLSB code because a listener is used. This listener must know where to listen. This information is declared in the *ActivationSpec*. The *activationSpec* is one of the accompanying elements of the MDB. For each MDB an *activationSpec* must be defined by the administration of WebSphere Application Server. It is declared under the TM Resource Adapter provider.

A.5 Feed from an IMS application in MashupHub

The code in A-5 is “ASIS”.

Example: A-5 Example of an IMS Feed

```

<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>https://server.company.com:9443/mashuphub/client/plugin/generate/
entryid/304/pluginid/7</id>
  <subtitle type="text">A list of insurance policy holders</subtitle>
  <link href="https://server.company.com:9443/mashuphub/client/plugin/generate/
entryid/304/pluginid/7" rel="self" type="application/atom+xml"></link>
  <updated>2008-05-28T21:10:14.189Z</updated>
  <title type="text">Insurance Policy Holders</title>
  <generator>InfoSphere MashupHub</generator>

  <entry xmlns="http://www.w3.org/2005/Atom">
    <title type="text">Item 1</title>
    <id>urn:uuid:1</id>
    <updated>2008-05-28T21:10:14.189Z</updated>
    <author>

```

```

    <name>authorname@company.com</name>
  </author>
  <summary type="text">Atom Feed entry 1</summary>
  <content type="application/xml">
    <RepeatingElement
xmlns="http://www.ibm.com/xmlns/atom/content/imsrecord/1.0">
      <POLICYDATA>
        <out_ll>964</out_ll>
        <out_zz>512</out_zz>
        <policy_detail>
          <policyid>AM4470</policyid>
          <customername>Alva Morua</customername>
          <coverage>104021.00</coverage>
          <premium>184.00</premium>
          <startdate>2006-10-23</startdate>
          <enddate>2007-05-23</enddate>
          <description>For physical loss or damage, coverage includes the hull,
machinery, fittings, furnishings, and permanently attached equipment for an agreed
value. These policies also provide broader liability protection than a
homeownerspolicy.</description>
          <address>11151 pine st</paddress>
          <city>Dixie</city>
          <state>Florida</state>
        </policy_detail>
        <policy_detail>
          <policyid>CG7010</policyid>
          <customername>Cruz Gorell</customername>
          <coverage>451171.00</coverage>
          <premium>775.00</premium>
          <startdate>2006-09-18</startdate>
          <enddate>2006-12-18</enddate>
          <description>The policy coverage provides compensation liability for
injury to persons employed by you who may work on your boat, but are not crew
members. Mechanics, carpenters, painters, and cleaners are included in this
category.</description>
          <address>11285 riverside dr</paddress>
          <city>Arkansas</city>
          <state>Arkansas</state>
        </policy_detail>
        <policy_detail>
          <policyid>CS5000</policyid>
          <customername>Clark Sandigo</customername>
          <coverage>363580.00</coverage>
          <premium>1395.00</premium>
          <startdate>2006-09-20</startdate>
          <enddate>2008-04-20</enddate>
          <description>The policy covers physical damage to the hull, sails,
machinery, furniture, and most other equipment that is normally used on board.
Most perials results from latent defects of workm</description>
          <address>63979 tree blvd</paddress>
          <city>Presidio Valley</city>
          <state>Texas</state>
        </policy_detail>
        <policy_detail>
          <policyid>CS8416</policyid>
          <customername>Chase Saucedo</customername>
          <coverage>424210.99</coverage>
          <premium>289.00</premium>
          <startdate>2006-10-19</startdate>
          <enddate>2008-03-19</enddate>

```

```

        <description>The policy coverage provides compensation liability for
injury to persons employed by you who may work on your boat, but are not crew
members. Mechanics, carpenters, painters, and cleaners are included in this
category.</description>
        <address>82814 oak st</address>
        <city>Lafayette</city>
        <state>Arkansas</state>
    </policy_detail>
    <policy_detail>
        <policyid>FC2267</policyid>
        <customername>Foster Cadaviec</customername>
        <coverage>310821.00</coverage>
        <premium>219.00</premium>
        <startdate>2006-10-27</startdate>
        <enddate>2007-04027</enddate>
        <description>For physical loss or damage, coverage includes the hull,
machinery, fittings, furnishings and permanently attached equipment for an agreed
value. These policies also provide broader liability protection than a homeowners
policy.</description>
        <address>38356 tree blvd</address>
        <city>Dawson</city>
        <state>Florida</state>
    </policy_detail>
</POLICYDATA>
</RepeatingElement>
</content>
</entry>
</feed>

```

A.6 WSDL files for a DLIModel generated Web Services Service

The code in A-6 is "ASIS".

Example A-6 shows a DLIModel-generated wsdl.

Example: A-6 DealersModels.wsdl

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:ims="http://www.ibm.com/ims/das/config" xmlns="http://ibm.sj.dealership"
targetNamespace="http://ibm.sj.dealership">
<!--Types-->
<wsdl:types>
<xsd:schema targetNamespace="http://ibm.sj.dealership">
<import xmlns="http://www.w3.org/2001/XMLSchema" schemaLocation="AUTPSB11-AUTS2PCB.xsd"
namespace="http://www.ibm.com/ims/AUTPSB11/AUTS2PCB"/> // <----- pointer to Schema Location see Example
16-4
<xsd:element name="getDealerModels">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="DLRNO">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:length xmlns="http://www.ibm.com/ims/das/config" value="4"/>
</xsd:restriction>
</xsd:simpleType>

```

```

</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="getDealerModelsResponse">
<xsd:complexType>
<xsd:sequence>
<element xmlns="http://www.w3.org/2001/XMLSchema" xmlns:pcb="http://www.ibm.com/ims/AUTPSB11/AUTS2PCB"
ref="pcb:MODEL" minOccurs="0" maxOccurs="1"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
</wsdl:types>
<!--Messages-->
<wsdl:message name="getDealerModelsRequest">
<wsdl:part name="parameters" element="getDealerModels"/>
</wsdl:message>
<wsdl:message name="getDealerModelsResponse">
<wsdl:part name="parameters" element="getDealerModelsResponse"/>
</wsdl:message>
<!--Ports-->
<wsdl:portType name="DealerModels">
<wsdl:operation name="getDealerModels">
<wsdl:input name="getDealerModelsRequest" message="getDealerModelsRequest"/>
<wsdl:output name="getDealerModelsResponse" message="getDealerModelsResponse"/>
</wsdl:operation>
</wsdl:portType>
<!--Bindings-->
<wsdl:binding type="DealerModels" name="DealerModelsSoapBinding">
<wsaw:UsingAddressing xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" wsdl:required="false"/>
<wsdlsoap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
<wsdl:operation name="getDealerModels">
<wsdlsoap:operation soapAction="getDealerModels"/>
<wsdl:input name="getDealerModelsRequest">
<wsdlsoap:body use="literal"/>
</wsdl:input>
<wsdl:output name="getDealerModelsResponse">
<wsdlsoap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<!--Services-->
<wsdl:service name="DealerModels_Services">
<wsdl:port name="DealerModels" binding="DealerModelsSoapBinding">
<wsdlsoap:address location="http://localhost:9080/DealerModels/services/DealerModels"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

A.7 XSD files for a DLIModel generated Web Services Service

Example A-6 on page 364 references the next xsd files in Example A-7.

Example: A-7 AUTPSB11-AUTS2PCB.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ims="http://www.ibm.com/ims"
  xmlns="http://www.ibm.com/ims/AUTPSB11/AUTS2PCB"
  targetNamespace="http://www.ibm.com/ims/AUTPSB11/AUTS2PCB"
  elementFormDefault="qualified">

  <xsd:annotation>
    <xsd:appinfo>
      <ims:DLI version="2.0" mode="retrieve" topLevelElement="AUTS2PCB" />
    </xsd:appinfo>
  </xsd:annotation>

  <!-- AUTS2PCB -->
  <xsd:element name="AUTS2PCB">
    <xsd:annotation>
      <xsd:appinfo>
        <ims:pcb name="AUTS2PCB" alias="AUTS2PCB"/>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="DEALER" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <!-- DEALER -->
  <xsd:element name="DEALER">
    <xsd:annotation>
      <xsd:appinfo>
        <ims:segment name="DEALER" alias="DEALER"/>
      </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="DLRNO" >
          <xsd:simpleType>
            <xsd:annotation>
              <xsd:appinfo>
                <ims:field name="DLRNO" alias="DLRNO"/>
              </xsd:appinfo>
            </xsd:annotation>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="4"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="DLRNAME" >
          <xsd:simpleType>
            <xsd:annotation>
              <xsd:appinfo>
                <ims:field name="DLRNAME" alias="DLRNAME"/>
              </xsd:appinfo>
            </xsd:annotation>
          </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

```



```

        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="30"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="CITY" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="CITY" alias="CITY"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="10"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="ZIP" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="ZIP" alias="ZIP"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="10"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="PHONE" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="PHONE" alias="PHONE"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="7"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
    <xsd:element ref="MODEL" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<!-- MODEL -->
<xsd:element name="MODEL">
    <xsd:annotation>
        <xsd:appinfo>
            <ims:segment name="MODEL" alias="MODEL"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="MODKEY" >
                <xsd:simpleType>
                    <xsd:annotation>
                        <xsd:appinfo>

```

```

        <ims:field name="MODKEY" alias="MODKEY"/>
    </xsd:appinfo>
</xsd:annotation>
<xsd:restriction base="xsd:string">
    <xsd:maxLength value="24"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="MODTYPE" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="MODTYPE" alias="MODTYPE"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="2"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="MAKE" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="MAKE" alias="MAKE"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="10"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="MODEL" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="MODEL" alias="MODEL"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="10"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="YEAR" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="YEAR" alias="YEAR"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="4"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="MSRP" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>

```

```

        <ims:field name="MSRP" alias="MSRP"/>
    </xsd:appinfo>
</xsd:annotation>
<xsd:restriction base="xsd:string">
    <xsd:maxLength value="5"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="COUNT1" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="COUNT" alias="COUNT1"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="2"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element ref="STOCK" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<!-- STOCK -->
<xsd:element name="STOCK">
    <xsd:annotation>
        <xsd:appinfo>
            <ims:segment name="STOCK" alias="STOCK"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="STKVIN" >
                <xsd:simpleType>
                    <xsd:annotation>
                        <xsd:appinfo>
                            <ims:field name="STKVIN" alias="STKVIN"/>
                        </xsd:appinfo>
                    </xsd:annotation>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="20"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="COLOR" >
                <xsd:simpleType>
                    <xsd:annotation>
                        <xsd:appinfo>
                            <ims:field name="COLOR" alias="COLOR"/>
                        </xsd:appinfo>
                    </xsd:annotation>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="10"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="PRICE" >
                <xsd:simpleType>

```

```

        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="PRICE" alias="PRICE"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="5"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="LOT" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="LOT" alias="LOT"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="10"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="WRNTY" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="WRNTY" alias="WRNTY"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="1"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>

```

Abbreviations and acronyms

ACEE	Access Control Environment Element	JCA	Java EE connector architecture
AGN	Application Group Name	JCL	Job Control Language
AIB	Application Interface Block	JDBC	Java Database Connectivity
APF	Authorized Program Facility	JDK	Java Development Kit
API	Application Programming Interface	JMP	Java Message Program
APPC	Advanced Program-to-Program Communication	JRE™	Java Runtime Environment
BPE	Base Primitive Environment	JSP	Java Server Pages
CCF	Common Connector Framework	JVM	Java Virtual Machine
CBM	Component Business Modeling	KBLA	Knowledge-Based Log Analysis
CGI	Common Gateway Interface	LAN	Local Area Network
CICS	Customer Information Control System	LPAR	Logical Partition
CSM	Complete Status Message	LTERM	Logical Terminal
CTG	CICS Transaction Gateway	LU	logical unit
DB2	Database 2	LU2	logical unit 2
DBCTL	Database Control	MCI	message control information
DBD	Database Description	MFS	message format services
DBRA	Database Resource Adapter	MOD	message output descriptor
EAB	Enterprise Access Builder	MPP	message processing program
ECB	Event Control Block	MSC	Multiple Systems Coupling
EJB	Enterprise Java Bean	MVS	Multiple Virtual System
EWLM	Enterprise Workload Manager™	ODBA	Open Database Access
GUI	graphical user interface	OLDS	online log data set
HTML	Hyper Text Markup Language	OM	Operations Manager
HTTP	Hyper Text Transfer Protocol	OO	object-oriented
IBM	International Business Machines Corporation	OTMA	Open Transaction Manager Access
IMS	Information Management System	OTMA C/I	OTMA Callable Interface
IPCS	Interactive Problem Control System	PC	personal computer
IPL	Initial Program Load	PCB	program communication block
IRM	IMS Request Message	PPT	Program Properties Table
ISC	Intersystem Communication	PSB	Program Specification Block
ISPF	Interactive Systems Productivity Facility	RACF	Resource Access Control Facility
ITSO	International Technical Support Organization	RAR	resource archive
IVP	Installation Verification Program	RM	Resource Manager
J2C	Java EE Connector Architecture	RMM	Request MOD Message
Java EE	Java Platform Enterprise Edition	RRS/MVS	Resource Recovery Services/MVS
JBP	Java Batch Program	RSM	request status message
		SCI	Structured Call Interface
		SGML	Standard Generalized Markup Language
		SMP/E	System Modification Program/Extended

SMU	Security Maintenance Utility
SNA	System Network Architecture
SOA	service-oriented architecture
SOMA	Service Oriented Modeling and Architecture
SOAP	Simple Object Access Protocol
SPOC	single point of control
SSPM	Sysplex serialized program management
STSN	Set and Test Sequence Numbers
SVL	Silicon Valley Laboratories
TCO	Time-Controlled Operations
TCP/IP	Transmission Control Protocol/Internet Protocol
TMRA	Transaction Manager Resource Adapter
TPIPE	Transaction pipe
UOR	Unit of Recovery
VIPA	Virtual IP Addressing
W3C	World Wide Web Consortium
WAN	wide area network
WLM	workload manager
WSDL	Web Service Definition Language
WWW	World Wide Web
XCF	Cross System Coupling Facility
XMI	XML metadata interchange
XML	Extensible Markup Language

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 377. Note that some of the documents referenced here might be available in softcopy only.

- ▶ *Powering SOA with IBM Data Servers*, SG24-7259
- ▶ *IMS Connectivity in an On Demand Environment: A Practical Guide to IMS Connectivity*, SG24-6794
- ▶ *IBM IMS Version10 Implementation Guide A Technical Overview*, SG24-7526
- ▶ *Monitoring WebSphere V5.1 Application Performance on z/OS*, SG24-6825-01

Other publications

These publications are also relevant as further information sources:

- ▶ *z/OS V1R9.0 Communications Server IP Configuration Guide*, SC31-8775-11
- ▶ *z/OS V1R6.0 Communications Server: IP System Administrator's Commands*, SC31-8781-04
- ▶ *HiperSockets Implementation Guide*, SG24-6816
- ▶ *IMS SOAP Gateway User's Guide and Reference Version 10 Release 1*, SC19-1290-01
- ▶ *IMS TM Resource Adapter User's Guide and Reference*, SC19-1211-02
- ▶ *IMS V10 Communications and Connections Guide*, SC18-9703
- ▶ *IMS V10 System Definition Guide*, GC18-9998
- ▶ *IMS V10 Database Administration Guide*, SC18-9704
- ▶ *IMS V 0 Database Utilities Reference*, SC18-9705
- ▶ *IMS V10 System Utilities Reference*, SC18-9968
- ▶ *IMS V10 Application Programming Guide*, SC18-9698
- ▶ *Business value of IMS and SOA: Better together July 2007*, IMW11876-UUSEN-00
- ▶ *Service-Oriented Architecture Compass: Business Value, Planning and Enterprise Roadmap*, ISBN:9780131870024
- ▶ *WebSphere MQ for z/OS System Setup Guide V6*, SC34-6583
- ▶ *IMS Connect Extensions for z/OS v2.1 User's Guide*, SC19-1163
- ▶ *IMS Java Guide and Reference Version 9*, SC18-7821
- ▶ *Security Administrator's Guide*, SA22-7683

Online resources

These Web sites are also relevant as further information sources:

- ▶ Gartner's Positions on the Five Hottest IT Topics and Trends in 2005 article
http://www.gartner.com/DisplayDocument?doc_cd=125868/
- ▶ Gartner Identifies the Top 10 Strategic Technologies for 2008 article
<http://www.gartner.com/it/page.jsp?id=530109>
- ▶ Component Business Modeling
http://www.ibm.com/services/us/gbs/bus/html/bcs_componentmodeling.html
<http://www.ibm.com/services/us/imc/pdf/g510-6163-component-business-models.pdf>
http://www.ibm.com/industries/financialservices/doc/content/bin/fss_bae_compone nt_business_modeling.pdf
- ▶ Download site for the runtime component of the IMS TMRA
<http://www-01.ibm.com/software/data/ims/ims/components/tm-resource-adapter.html#downloads>
- ▶ Service Oriented Modeling and Architecture (SOMA)
<http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/>
<http://www-935.ibm.com/services/us/gbs/bus/pdf/g510-5060-ibm-service-oriented-modeling-arch.pdf>
- ▶ IMS Info 2.0 demo video on YouTube
<http://www.youtube.com/watch?v=BWJGSC-RyXQ>
- ▶ Effective SOA governance, March 2006 article
<ftp://ftp.software.ibm.com/software/rational/web/whitepapers/soagov-mgmt.pdf>
- ▶ Enabling SOA through organization change and governance, November 2007, IBM White Paper
http://download.boulder.ibm.com/ibmdl/pub/software/solutions/soa/gov/GBS_S0Agov_Org_Change.pdf
- ▶ Rational Business Developer Extension executive overview
<http://www.ibm.com/developerworks/rational/library/may07/sergi/index.html>
- ▶ WebSphere Message Broker information site
<http://www-01.ibm.com/software/integration/wbimessagebroker/>
- ▶ WebSphere Business Monitor information site
<http://www-01.ibm.com/software/integration/wbimonitor/>
- ▶ IBM Infrastructure Healthcheck Services for SOA
<http://www.ibm.com/software/solutions/soa/healthcheck.html>
- ▶ WebSphere Application Server Community Edition Information site
<http://www-01.ibm.com/software/webservers/appserv/community/>
- ▶ WebSphere Application Server – Express information site
<http://www.ibm.com/software/webservers/appserv/express/>
- ▶ WebSphere Application Server information site
http://www.ibm.com/software/webservers/appserv/was/index.html?S_TACT=103BGW01&S_CMP=campaig

- ▶ WebSphere Application Server Network Deployment information site
http://www.ibm.com/software/webservers/appserv/was/network/index.html?S_TACT=105AD02W&S_CMP=campaign
- ▶ WebSphere Application Server for z/OS information site
http://www.ibm.com/software/webservers/appserv/zos_os390/
- ▶ WebSphere Application Server training resources
<http://www.ibm.com/software/websphere/education/>
- ▶ Demos and Videos on WebSphere Application Server
http://www.ibm.com/software/info/television/index.jsp?lang=en_us&cat=websphere&item=xml/U939990E82018I09.xml&S_CMP=rnav
- ▶ Rational Application Developer information site
http://www.ibm.com/software/awdtools/developer/application/?S_TACT=105AGX15&S_CMP=LP
- ▶ Information on Business Process Management
<http://www.ibm.com/software/info/bpm/>
http://www.ibm.com/software/integration/wdpe/features/?S_CMP=rnav
<http://www.ibm.com/software/data/content-management/products/process.html>
- ▶ Information on WebSphere Business Modeler
<http://www.ibm.com/developerworks/websphere/usergroups/>
<http://www.ibm.com/developerworks/websphere/zones/bpm/>
<http://www.redbooks.ibm.com/abstracts/redp4159.html?Open>
<http://www.redbooks.ibm.com/abstracts/redp4433.html?Open>
- ▶ Information on WebSphere Service Registry and Repository
<http://www.ibm.com/software/integration/wsrr/>
- ▶ Information on WebSphere Integration Developer
<http://www.ibm.com/software/integration/wid/>
- ▶ Information on IMS SOAP Gateway
<http://www.ibm.com/software/data/ims/soap/>
- ▶ IBM Enterprise Modernization Sandbox for System z: Architecture
http://www.ibm.com/developerworks/downloads/emsandbox/systemz/architecture/?S_TACT=105AGX15&S_CMP=EMDOC
- ▶ The IBM Tivoli OMEGAMON XE for Mainframe Networks Library
<http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon.mn.doc/sc32-1924-0009.htm>
- ▶ Information on Omegamon XE for IMS on z/OS
<http://www.ibm.com/software/tivoli/products/omegamon-xe-ims/>
- ▶ WebSphere InfoCenter (V6.1)
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/welcome_base.html
- ▶ Information on Rational ClearQuest
<http://www.ibm.com/software/awdtools/clearquest/index.html>
- ▶ Problem Determination Tools for System z

<ftp://ftp.software.ibm.com/software/http/pdtools/PD-Tools-Reference-Guide-Sept-2007-WS011258-USEN-01.pdf>

<http://www.ibm.com/software/awdtools/deployment/>

- ▶ Information on Fault Analyzer for z/OS
<http://www.ibm.com/software/awdtools/faultanalyzer/>
- ▶ Information on File Manager for z/OS
<http://www.ibm.com/software/awdtools/filemanager/>
- ▶ Information on Application Performance Analyzer for z/OS
<http://www.ibm.com/software/awdtools/apa/>
- ▶ Information on Workload Simulator for z/OS
<http://www.ibm.com/software/awdtools/workloadsimulator/>
- ▶ Information on the Debug Tool for z/OS
<http://www.ibm.com/software/awdtools/debugtool/>
- ▶ IMS Performance Analyzer for z/OS
<http://www-01.ibm.com/software/data/db2imstools/imstools/imspace.html>
- ▶ IMS Problem Investigator for z/OS
<http://www-01.ibm.com/software/data/db2imstools/imstools/imsprobleminvest.html>
- ▶ Information on the Eclipse Test and Performance Tools Platform (TPTP)
<http://www.eclipse.org/tptp/index.php>
- ▶ Understanding the Choices: IMS Connect and WebSphere MQ
<http://w3-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100638>
- ▶ TM Resource Adapter installation site
<http://www.ibm.com/software/data/ims/ims/components/tm-resource-adapter.html>
- ▶ RESTful Web services
<http://java.sun.com/developer/technicalArticles/WebServices/restful/>
- ▶ IBM Mashup Center
<http://www-01.ibm.com/software/info/mashup-center/>
- ▶ Eclipse based DLIModel Utility GUI plug-in
<http://www.ibm.com/software/data/ims/toolkit/dlimodelutility>
- ▶ XPath API
<http://www.ibm.com/developerworks/library/x-javxpathapi.html>
- ▶ IBM WebSphere DataPower SOA Appliances
<http://www.ibm.com/software/integration/datapower/index.html>
- ▶ IBM Developerworks article on XML security threats
http://www.ibm.com/developerworks/websphere/techjournal/0603_col_hines/0603_col_hines.html

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, and order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

A

accessor environment element (ACEE) 176
ACEE 94, 147, 176, 195
ACK 138, 148, 162, 170, 175–176, 182, 210, 212, 214, 216
ActivationSpec 238, 242, 358
Address space 328
address space 25, 167, 322–323
Advanced Functions (AF) 109
AIB 158, 239–240, 323
ALTPCB 151, 164–165, 186, 211, 213, 234–235, 237
APF 193
application interface block (AIB) 158, 239, 323
application program 322
application programming interface (API) 8, 14, 24, 49, 86, 131, 149, 169, 176, 294, 295, 299, 304, 321, 322, 330, 333
application resource 119
Application server 87, 225, 251, 265, 294, 346
 CPU utilization 134
 dynamic cache service 127
 successful connection 125
Architecture agility 42
ASCII 76–77, 180, 190
ASN.1 347
Asynchronous callout 146, 186, 209, 236, 357
 Code snippet 357
 Message flow 236
 web service operations 209
 Web Services callout scenarios 210
asynchronous hold queue
 name 213, 237
 SGTP1 TPIPE 211, 213
asynchronous output 93, 182, 199, 226, 232–233, 236
AT-TLS 97

B

Base Primitive Environment (BPE) 290
blog 21, 280
BPM xv, 4, 63
Business analyst 37, 42
business assets 27
business process 4, 7, 11, 27, 37, 40–42, 64–65, 106, 229
Business Process Management (BPM) 4, 63, 65
business service 39–40, 42
business value 16, 27, 32, 271

C

callout EJB 237
callout message 156–157, 160, 211, 234, 236–238, 240, 356
 OTMA headers 211, 213

XML transformation 216
callout request 151, 165, 186–187, 204–205, 225, 229, 233–235, 285
 OTMA routing 235
callout request message 187, 212, 237
 Web Service identifier 188, 209
CLI 348
client application 17, 72, 155, 167, 170, 175–176, 204, 207–208, 227, 240–241
 IMS TM resource adapter interactions 227
clientID 98, 164, 170, 232–233
COBOL 12, 14, 16, 49–50, 59, 67, 72, 110–111, 146, 157, 160, 172, 176, 205, 208, 281, 287–288, 301, 303
COBOL copybook 14, 206, 208, 227, 285, 303, 347
 additional field information 23
COBOL format 175
 IMS SOAP Gateway input message 175
command line
 development tool 256
 interface 348
 version 303
commit mode 98, 227, 285, 356
commit-then-send 147, 170, 199
Common Application Metamodel (CAM) 246
Common Programming Interface for Communications (CPIC) 152
Common Service Layer (CSL) 322, 329
Complete Status Message (CSM) 182
Component Business Modeling (CBM) 7, 43
Config member 101
configuration member 73, 95–96, 149, 171–172, 176, 290
Connection bundle 81, 100, 185, 204
 Property options 204
 file 100, 204
 property 100
Connection Factory 89, 98, 170, 183, 230, 265, 299, 357
connection pooling 128, 226, 228, 230, 265
ConnectionFactory 89, 236
conversational 45, 146, 176, 182, 249, 259, 285
conversations 9, 20, 152
converter name 165, 205, 210–211
CORBA security 86
correlator file 100, 185, 205, 283, 286–287
Coupling Facility 160
CSL 172, 198, 322, 330
CSS 20

D

data sharing xv
data transformation 83, 187, 206, 251
database xv, 14, 37, 59, 98, 110, 146, 196, 210, 235, 248, 274, 293–294, 321
database management system (DBMS) 333

- Database Resource Adapter (DRA) 197, 297, 322, 324
- databinding 66
- data-centric approach 311
- DataPower 26, 341
- DataPower appliance 345
 - performance advantage 346
- datastore 97, 99, 129, 171–172, 359
- DATASTORE statement 94, 96, 174
 - ID parameter 99
- DB resource adapter (DRA) 294, 322
- DB2 14, 16, 28–29, 63, 110–111, 130, 180, 294, 326
- DBCS 20
- DBCTL 12, 160, 171, 221, 294, 306
- DCCTL 12, 168–169
- Debug Tool (DT) 109
 - other interactions 111
- decomposed storage 311, 313–314
- dedicated hardware 346
- DELETE 286, 334
- Demand Environment 105, 145, 169
- dependent region 14, 152, 159, 163, 210, 296–298, 326–327
- descriptor DESTSG1 211
- destination name 165
- DFSYDRU0 165, 234, 236
- DFSYDTx 158, 165, 234
- DFSYPRX0 165, 234, 236
- Distributed Relational Database Architecture (DRDA) 149, 176, 300, 322
- distributed WebSphere Application Server
 - local accessor 298
- DL/I 29, 49, 151, 156, 186, 217, 296–298, 321, 323
- DL/I call 217, 321
- DLI data 293
 - database 298
 - Web Servers 307
- DLIModel 14, 16, 23, 146, 293, 295, 364, 366
- DLIModel Utility 23, 25, 293, 297, 299–300, 303
 - business goal 23
- Document Object Model (DOM) 318
- DRA 12, 14, 294–295, 322
- DRU 163, 165, 173–174
- DSN 80, 102

E

- EBCDIC 77, 180, 190, 208, 265, 290, 349
- ebXML 347
- ECB 101
- Eclipse Modeling Framework (EMF) 256
- Eclipse Test and Performance Tools Platform 130, 132
- EGCS 20
- EJB 59, 118, 156, 183, 198, 218, 226, 229, 265, 295, 297, 327
- EMH 201
- Enterprise backbone 4
- Enterprise Edition (EE) 14, 58, 225
- Enterprise Information System (EIS) 9, 18, 98
- Enterprise Java Bean (EJB) 28, 36, 119, 234, 295
- Enterprise Metadata Discovery (EMD) 227, 229, 263
- Enterprise Service Bus (ESB) 49

- Event collector 191, 193
- exit 95, 149–150, 176, 222, 235–236, 259, 290, 334, 337, 349–350
- exit routine 149–151, 198
 - query information 150
 - relevant data 151
- Extensible Markup Language (XML) 246, 281, 301, 342
- external application 28, 43, 146, 151, 155–156, 186–187, 209, 216–217, 229, 235, 239
 - Send-Only message processing functions 217
- external partners 294
- External Subsystem Attach Facility (ESAF) 154, 161

F

- Fault Analyzer 59, 109
- feed 7, 271, 274, 279
- feed mashup 7, 286, 292
- File Manager (FM) 109
 - IMS component 110
- FMID 296
- Front End
 - Shared queue environments 221

G

- governance framework 38, 343
- graphical user interface (GUI) 106, 111, 303, 343
- GroupName property 99

H

- HALDB 312, 322
- HDAM 312
- HIDAM 312
- hierarchical organized DLI data
 - XML exploitation 319
- high-performance XSL 343
- Hipersockets 123
- hold queue 95, 175, 211, 213, 234–235
 - callout request 236
 - output messages 95
- HOSTNAME 101, 173
- HTML 11, 14, 59, 91, 342
- HTTP 9, 17, 53, 90–91, 118, 187, 204, 251, 279, 310, 345–347
- HTTP request 91, 127, 251
 - state information 260
- HTTP session 251
- HWS 94–96, 172–173
- HWSCFGxx 94, 96, 198, 200, 222
- HWSCSLO0 172, 180
- HWSCSLO1 172, 180
- HWSIMSO0 101
- HWSJAVA0 176
- HWSP1410W message 178
- HWSSMPL0 96, 101, 172–173, 176
- HWSSMPL1 96, 101, 172, 175–176
- HWSYDRU0 173–174

- I
- IBM FileNet Active Content Edition 63–64
- IBM IMS 2, 143, 145–146, 188–189, 285
 - tool 193
- IBM Mashup Center Enterprise Edition 22, 281
- IBM MQseries status (IMS) 105, 129, 145, 167, 199–200
- IBM Tivoli Composite Application Monitor (ITCAM) 133
- ICAL call 158, 240
- ICAL SENDRECV
 - call 159, 188, 217–218
 - synchronous call function 239
- illustrate a more simplistic (IMS) 321, 347
- IMS xv, 1, 3, 17, 27–28, 57–58, 69, 71–72, 105, 118, 143, 145, 167–168, 203, 225, 245, 271, 293–294, 321, 339, 341, 2
 - parameter index 289
- IMS application 12, 14, 16–17, 27, 35, 81, 129, 145–146, 151, 155, 161, 167, 175–176, 185, 204, 225–226, 235, 245–246, 282, 285–286, 296–297, 321
 - alternate PCB destinations 188
 - alternate TPPCB 209
 - App1 211
 - App2 214
 - AppI2 214
 - callout messages 157, 188, 217
 - callout processing 160
 - data format 210
 - data structure 208, 286
 - developer 286
 - development 23
 - external Java EE applications 155
 - format response 207
 - format transformation 207
 - inbound and outbound message structure 287
 - instance 185–186, 209
 - issue 237
 - output data structure 22, 286
 - program 158, 175, 188, 216, 235
 - program issue 218
 - programming style 217
 - source file 287
- IMS asset 11, 204, 273
- IMS command 93, 201
- IMS Connect 14, 28–29, 72, 83, 88, 105, 108, 129, 147, 155, 167, 171, 204, 226–228, 248, 250, 271, 283, 285, 297, 322, 327, 347–348, 359
- IMS Connect commands 94, 196
- IMS Connect Extensions 15, 29, 105, 129, 167, 188
- IMS Connect security 95, 97
- IMS Connect SSL 83, 97
- IMS Connector 160, 225
- IMS Connectivity 169
- IMS Connector for Java 18, 232
- IMS Control Center 172, 179–180, 181
- IMS conversational transaction 232
- IMS data xv, 16, 25, 32, 77, 110, 149, 177, 198, 273, 297–298, 311, 313, 316, 322, 326, 328–329, 2
- IMS database 14, 23, 110, 296–297, 301, 316, 321, 326
 - annotated XML schema 302
 - hierarchical structure 316
 - intact XML documents 315
 - segment occurrences 110
 - SQL support 297
 - store XML data 23
 - XML document 315
 - XML documents 146
 - XML storage 312
- IMS database resource
 - adapter 325
 - distribution 25
- IMS Datastore (ID) 86, 91, 173
- IMS DB 28, 149, 296, 321–322
 - TCP/IP path 176
- IMS destination (ID) 349
- IMS DLI
 - language 318
- IMS host
 - application 28, 251
 - information 291
- IMS Info
 - 2.0 service adapter 287
- IMS Info 2.0
 - project 286
 - service adapter 287
- IMS Java 147, 179, 296–297, 326–327
- IMS MFS Web Enablement 19–20, 249
- IMS Open Database 24, 149, 172, 322
 - Access 28, 298
 - Manager 176
 - support 322
- IMS OTMA 83, 88, 95, 99–100, 160, 172, 218, 233, 236
 - Asynchronous Queue 232–233
 - group 83, 172
 - member name 99–100, 161
- IMS Performance Analyzer 129–130, 139, 193, 196–197
- IMS Problem Investigator 130, 191, 193, 197
- IMS record 316
- IMS Request Message (IRM) 210
- IMS service 13, 17, 30–31, 69, 71, 98, 117, 179
 - securing and tracking 85
- IMS SOA xv–xvi, 2, 16, 35, 58, 141, 143, 145, 176, 179, 292, 295, 352, 2
 - component 53
 - Composite Business Application support 146
 - enhancement 166
 - implementation 57–58
 - implementation project 40
 - Integration xvi, 16
 - Integration Suite 16, 18
 - project 49, 58, 63
 - solution 148
- IMS SOA Integration suite
 - security 98
- IMS SOA supporting tools 63
- IMS SOAP Gateway 16–17, 28, 71–72, 100, 104, 146, 155, 172, 175–176, 203, 292
 - 10.1 216
 - 10.2 222
 - Asynchronous Callout 217
 - asynchronous callout mode 187

- callouts 187
- Deployment Utility 204, 222
- feature 207
- high-level understanding 203
- message 175
- Multiple segment support 224
- pre-deployed Web Service 2 212
- RESUME TPIPE request processing 215
- second Web Service 212
- server 188, 206
- solution 207
- Synchronous Callout 217
- synchronous callout mode 188
- tool 204
- Version 18
- Version 10.1 159
- Web Services 185, 187
- XML conversion support 165
- XML response message 210
- IMS SOAP gateway 14, 104, 155, 175, 204
 - Web services 17
- IMS subsystem 168, 198, 294, 300, 329–331
 - different LPAR 331
- IMS system 12, 18, 59, 108, 151, 161, 171, 185, 210, 225, 248, 299, 322, 325
- IMS TM 12, 14, 28, 88, 146, 172, 176, 217, 225, 248, 263, 292, 325
- IMS TM Resource Adapter 14, 16, 98, 114, 146, 170, 185, 227, 229, 249, 263
 - security 98
- IMS tooling 196
- IMS Transaction Manager 12, 225–226, 243, 306
- IMS transaction 13–14, 16, 18, 28, 53, 66, 72, 83, 92, 100, 125, 155–156, 160–161, 183–184, 188, 205, 209, 225–226, 229, 232, 247–248, 273–274, 297, 322, 327, 350
 - code 208, 214, 265
 - data 83, 218, 238, 252, 273
 - look 185
 - traditional monitoring requirements 53
- IMS Universal DL/I driver 329, 331
- IMS Universal Driver
 - IMS database resource adapters 322, 325
- IMS Universal Drivers 148, 149, 177, 321
- IMS Universal JDBC driver 300, 329, 331
- IMS V10
 - Communication 97, 160, 169
 - System Definition Guide 169
- IMS Version 10 22, 83, 96, 145–147, 175–176, 209, 229, 233, 267–268, 271, 285, 296, 303
 - 10 Communication 236
 - 10 feature 146
 - 10 Implementation Guide 175
 - 10 maintenance 148
 - 10 SPE 297
 - 10 system 148
 - 10 user 176
 - 11 24–25, 145, 148, 176–177, 233, 297, 299, 326
 - 11 DB resource adapter 327
 - 11 OTMA function 163

- 11 QPP 224
- 7 323
- 8 196
- 9 15, 148, 168, 175, 208, 250, 267, 297, 316
- V9.1 263
- IMS Web 2.0
 - solution 21, 23, 273
 - Solution service adapter 284
 - Solutions architecture 283
- IMSAMEM,DRU (ID) 173, 257
- IMSCON TMEMBER 159
- IMSCONN.CERT Bin 102
- IMSConnectionSpec 236, 249, 336
- IMSPLEX 172, 198
- IMSplex 25, 138, 168, 171, 180, 199, 322, 324–325
- environment 180
- include configuration member support (IMS) 145, 167, 189, 201
- Information lifecycle management (ILM) 106
- Information Management Software (IMS) 271–272, 321
- Information Management System (IMS) 3
- Information Technology Infrastructure Library (ITIL) 105
- InfoSphere MashupHub 21–22, 271, 281–282
- input message 72, 146, 161, 163, 175, 199, 214, 265
 - maximum number 146, 165
- input message successfully (IMS) 204
- input parameter 240, 284
- input/output message
 - definition 14
- intact storage 312, 314
- Integrated Development Environment (IDE) 18, 226, 264
- integrated IMS 22, 149, 286
 - IMS Version 10 286
- InteractionSpec 231–232, 237, 268, 270, 360
- Internet xv, 18–19, 82, 85, 172, 249–250, 272, 274, 327, 345–346
- Internet Protocol
 - Version 4 172
 - Version 6 172
- introduce multi-segment (IMS) 203, 233, 245, 271–272
- IOPCB 152, 186, 228
- IP address 9, 98, 100, 219, 301, 343, 348–349
- IPV6 172
- ISO 8583 347
- ISPF 78–79, 110, 139, 180, 189
- ISPF dialogue 188
- ISRT ALTPCB 151, 186, 237
 - call 187, 238
- ITCAMSE for WDP 348
- ITIL 105
- IVP 71, 82, 258
- IVTNO 258, 288

J

- J2C 18, 118, 183–184, 225–227, 262
- J2C Java bean 227, 262, 265
- J2EE 14, 17, 37–38, 59, 86, 114, 126, 183, 203, 225–226, 249, 256, 298, 321, 325, 327, 359
- J2EE application 87, 130, 238, 298
- J2EE artifact 229

J2EE Connector Architecture (J2C) 18, 262, 330
 J2EE container 229, 327, 330
 Java xv, 14, 16, 28–29, 36, 59, 79, 86, 106, 108, 114, 126, 147, 155, 177, 181, 204, 225, 248, 251, 293–295, 321, 326–327
 Java application 18, 24, 96, 114, 133, 198, 225, 229, 235, 265, 296–297, 301, 304, 322, 325, 327, 331
 Java applications 18, 62, 225, 297, 326, 328–329, 331
 Java batch processing (JBP) 326
 Java Batch Program (JBP) 294
 Java class
 library 296, 335
 Java dependent region (JDR) 294
 Java development 296
 Java Message Program (JMP) 294
 Java Server Page (JSP) 90, 183
 Java Virtual Machine (JVM) 118
 JBP 14, 160, 217, 221, 240, 294, 296–297, 326
 JCA 14, 226, 262, 295, 325, 327
 JCL 59, 95, 290
 JDBC 14, 24, 28, 45, 139, 146, 294, 297–298, 321, 326
 JDBC driver 296–297, 300, 321, 325–326, 329, 331
 XQuery support 318
 JMeter 130, 138
 JMP 14, 217, 240, 294, 296–297, 326
 JVM 14, 81, 86, 118

K

Keyed Sequence Data Set (KSDS) 59, 189
 keystore 98, 100

L

Language Environment 129
 line-of-business (LOB) 37
 Linux 108, 123, 131, 180, 226
 LOCK and UNLOCK (LU) 146
 logical relationships 295, 302
 Lotus Mashup 282–283, 292
 LPAR 322

M

mashup 7, 22, 271, 273, 280
 MAXSOC 101, 173–174
 MDB 28, 156–157, 186, 218, 229, 235
 Mediation Module 184, 229
 Message Driven Bean (MDB) 185, 235
 Message Format Service (MFS) 16, 19, 59, 245
 Message Format Service SOA support 262
 Message Input Descriptors (MID) 252
 message processing program (MPP) 185
 message-driven bean (MDB) 235
 metadata 15, 23, 65, 247, 249, 280, 282, 293, 295, 343, 347
 MFS xvi, 14, 44, 67, 100, 146, 182, 227, 245, 273–274, 285
 MFS Common Application Metamodel 246, 256
 MFS Importer 246
 MFS servlet 251

MFS SOA Solutions 21
 MFS source
 file 229, 246
 file result 256
 relationship 246
 MFS Web Enablement xvi, 19–20, 245, 248–249
 list support 261
 MFS Adapter component 252
 MFS Web Solutions 245–246
 middleware 53, 55, 61, 130, 133, 341, 344
 MOD 182–183, 252
 mode 76, 98, 110–111, 124, 154, 180, 182, 226–227, 256, 285, 298, 313, 331, 356
 mode 0 98, 210, 227
 MPP 14, 185, 217, 240, 296
 MQSeries 161
 MS SOAP Gateway
 security 100
 MSDB 302
 MSMQ 9
 multi-layer framework 329
 multiple IMS
 application 148, 219
 applications Scenario 2 212
 DB system 148
 system 148, 199
 MVS 73, 131, 256, 330

N

NAK negative acknowledgement 138, 162–163, 182, 210
 NAK response 162, 182
 non-z/OS platform 327, 331

O

object code only (OCO) 180
 ODBA 14, 28, 148, 197, 297–298, 322, 328
 ODBA interface 148, 298, 324
 ODBA module 323
 OM 168, 172, 180, 200, 325
 OMEGAMON 15, 106, 108, 138, 193, 197
 OMEGAMON XE 106, 138, 197
 on demand xv, 2
 On Demand and SOA 146
 Open Data Base Access (ODBA) 198
 Open Data Base Adapter (ODBA) 148
 Open Data Base Manager (ODBM) 297, 322
 Open Database
 enhancement 149
 Manager 25, 149, 172, 322, 329
 open systems interconnection
 ISO reference model 86
 open systems interconnection (OSI) 86
 Open Transaction Manager Access (OTMA) 146, 155, 160, 161, 172, 175, 233
 operating system (OS) 9
 Operations 172, 193
 operations 9, 64, 92, 105, 119, 126, 167, 206, 262, 279, 336
 Operations Manager (OM) 168, 200

OTMA 36, 83, 88, 92, 105, 129, 145–146, 171, 175–176, 211, 232–233, 350
 descriptors 147, 164, 234
 transaction pipe 161, 199
 OTMA ALTPCB descriptor 165
 OTMA callout security 94
 OTMA client 92, 94, 146–147, 161–162, 199–200, 238, 241–242
 associates application output 161
 certain characteristics 165
 descriptor 165
 descriptor entry 165
 front-end IMS 200
 NAK response 165
 new messages 162
 other communications 147
 security levels 163
 OTMA descriptor 155, 157–158, 164, 210, 218, 234, 240
 entry 164
 name 156, 188, 218, 240
 OTMA Destination
 Resolution exit routine 165, 236
 OTMA protocol
 command 163
 update 240
 OTMA security 92, 147, 163, 236
 OTMA super member 165, 199
 function 199
 tpipe 199
 output message 9, 14, 44, 72, 83, 93, 147, 156, 165, 175, 208, 216, 228, 236, 246, 254, 265
 output XML 72, 208, 291

P

Parallel xv
 Parallel Sysplex xv
 password length 100–101
 password-protected database 98
 PCB 23, 164, 188, 300, 336
 performance xv, 1, 29, 31–32, 59, 69, 93–94, 106, 108, 117, 160, 170, 279, 298, 332, 341, 343
 performance data 118
 Performance measurement infrastructure (PMI) 118
 performance monitoring (PM) 53, 131
 Performance monitoring tools 130
 persistent socket 98, 170, 182
 PK24912 83
 PK29938 83
 PK37758 148
 PK37905 148
 PK38197 148
 PK41554 148
 PK42286 148
 PK43685 148
 PK66020 148
 PK66022 148
 PK70078 159
 PK70330 159
 PL/I 14, 16, 22, 44–45, 49, 59, 110–111, 146, 157, 160, 176, 221, 281, 285, 287

PL/I application 146
 port 10, 97–98, 108, 121, 172, 204, 233, 267, 291, 343, 349, 365
 PORTID 101, 173–174
 Practical Guide 105, 145, 169
 process model 34, 42
 PROCLIB 80, 151, 158, 165–166, 234, 290
 Program Call (PC) 168, 323–324, 328
 program communication
 block 308
 program control block (PCB) 336
 Program Properties Table (PPT) 172
 program specification block (PSB) 23, 298, 301, 330, 336
 programming model 16, 217, 325, 330
 project xvi, 33, 63, 66, 106, 283, 286, 303, 307, 310
 public class
 MDB4IMSASyncCallOut1Bean 357
 MDB4IMSSyncCallOut1Bean 361
 SLNB4IMSASyncCallOut1Bean 356
 SLNB4IMSSyncCallOut1Bean 360
 public key
 infrastructure 86

Q

Quality of Service (QOS) 345
 Quality Partnership Program (QPP) 224
 Queued Direct I/O (QDIO) 169

R

RACF 20, 73, 78, 92, 173–174, 176, 222, 235, 249, 258, 298, 330, 350
 RACF FACILITY class 95
 RACF Profile 93, 95
 RAD 14, 21, 58
 RAR 226, 299
 Rational Application Developer (RAD) 14, 18, 58, 126, 146, 155, 157, 159, 183, 206, 226–227, 251, 263, 267, 310
 J2C framework 262
 Rational Developer 18, 22, 28, 58–59, 72, 74, 109, 111, 146, 155, 157, 176, 187, 208, 221–222, 285–287
 Language Debugger component 111
 Rational Developer for zSeries (RDZ) 206
 Rational Performance Tester 130, 138
 Rational Performance Tester (RPT) 138
 Rational Software
 Architect 18, 106, 226
 Rational Unified Process (RUP) 33
 RDz 146, 176, 205, 283
 Redbooks Web site 377
 Contact us xvii
 Request message 210
 Request MOD Message (RMM) 182
 Request Status Message (RSM) 181
 Resource 8, 12, 14, 38, 40, 57, 87, 120, 130, 133, 146, 155, 164, 170, 172, 176, 225–227, 248, 250, 252, 279, 292, 294–297, 321–323, 330, 358
 Resource Access Control Facility (RACF) 86, 291
 Resource Adapter 14, 16, 88, 98, 114, 146, 170, 176,

- 179, 229, 249, 263, 294, 297, 323–324, 362
- Resource Measurement Facility 131
- response message 22, 156, 158, 171, 175, 209–210, 240–241, 286–287, 357
- RESUME TPIPE 94, 156, 182, 199, 212, 235, 238, 241–242
 - call 199
 - request processing 213
- Resume TPIPE 93, 175, 212
 - added support 175
 - command 94
 - loop 188, 209
 - request 237
- Resume Tpipe (RT) 155
- Roll Your Own (RYO) 179
- RRS 101, 198, 298, 323–324, 330

S

- SAF 92–93, 138
- same IMS 185, 217–219, 233
 - database 314
 - transaction instance 216
- scheduling 238, 242
- SCI 160, 168, 172, 200, 323, 330
- Secure Association Service (SAS) 86
- Secure Sockets Layer (SSL) 97
- security 18, 20, 35, 37, 53, 69, 79, 83, 85, 86, 126, 147–148, 167, 172, 176, 195, 204, 208, 226, 230, 235, 249, 279, 291, 298, 330, 327, 339, 341–343
- Security Attribute Service 87
- security authorization facility (SAF) 93, 99
- send-then-commit 147, 162, 170
- Service Component Architecture (SCA) 8, 229
- service data 212, 214, 216
- service level agreement 35, 129
- Service-oriented architecture (SOA) 1, 3–4, 8, 27, 59, 63, 168, 203, 276, 328, 341
- Service-oriented modeling
 - and Architecture 43
- Service Oriented Modeling and Architecture (SOMA) 34
- Service Provider Interface (SPI) 327
- Service-oriented (SO) 8
- Service-Oriented Architecture
 - Compass 37, 40
 - provides 11, 203
- servlet 59, 90, 119, 127, 249, 251
- SETRACF 95
- Shared Queues 29, 220, 240
- shared queues xv, 199–200
- Simple Object Access Protocol (SOAP) 8, 203
- single sign-on (SSO) 91
- SMP/E 296
- SMTP 9
- SNA terminal 165
- SOA environment 3, 16, 30, 53, 124, 294
 - IMS capabilities 16
 - IMS databases 294
 - reuse offer great value 30
- SOA solution 3–4, 29–30, 51, 121, 143
- SOA solutioning

- important role 160
- SOAP 8–9, 17, 28, 38, 71, 73, 100–101, 126, 139, 146, 151, 155, 175–176, 203, 279, 310, 343
- SOAP Gateway 16–17, 28, 49, 53, 71–72, 100–101, 104, 146, 155, 172, 175–176, 185, 203, 292
 - Multi-segment handling 222
- SOAP Gateway Connection bundle 100
- SOAP message 16, 72, 207, 210
- sockets 97–98, 123, 146, 151, 170, 227
- SPOC 180, 201
- SQL 11, 14–15, 295, 304, 325–326
- SSH 348
- SSL port 97–98, 172
- SSL support 83, 97, 204
- SSL Truststore
 - name 100, 204
 - password 101
- SLENVAR 101
- SSLPORT 101
- StateLess Session Bean (SLSB) 185–186, 234
- status monitor (SM) 193, 196
- storage 55, 178, 311
- Structured Call Interface (SCI) 168, 200, 330
- structured query language (SQL) 321
- super member 165, 199
- SXPL 151
- Synchronous Callout
 - Code snippet 360
 - Message flow 241
- synchronous callout 49, 145, 186, 216, 240, 360
 - call 157, 160, 186, 216, 221, 229, 240
 - data transformation 188
 - function 156–157, 188
 - message 157, 160, 186, 188, 217, 240–241
 - request 151, 158, 188, 217–218, 239–240
 - solution 210
 - support 156–157, 188, 216, 220
- synchronous callout request
 - message 160, 188, 217–218, 241
 - status 217
- synchronous callout support 155–156, 159, 187
- Sysplex xv, 130–131, 199, 201, 219
- Sysplex Distributor 199, 219
- System Authorization Facility (SAF) 86
- System z 1, 18, 28, 58, 71–72, 86, 109, 155, 157, 159, 168, 176, 221–222, 226, 285–287
 - IBM Rational Developer 58–59, 111
 - Rational Developer 18, 23, 58, 60, 159, 287
 - WebSphere Developer 18
 - WebSphere Developer Debugger 111
- System.out.println 334

T

- target IMS
 - job name 99
- TCP 9, 97–98, 108, 110, 121, 151, 159, 168, 287, 297, 322, 325
- TCP/IP 11, 14, 38, 53, 97–98, 108, 110, 121–122, 153, 155, 160, 168, 207–208, 212, 233, 273, 300, 322, 327, 329

- TCP/IP client 171, 201
 - application 155
- TCP/IP clients 15, 129, 171
- TCP/IP communication 170
- TCP/IP connection 98, 153, 161, 171, 213
- TCP/IP port 171, 197
- Telnet 110, 348
- TIMEOUT 101, 173–174
- timeout 156, 232–233, 237, 356
- Tivoli Access Manager WebSEAL 91
- Tivoli Composite Application Monitor 130, 133
- Tivoli Omegamon XE for IMS Connect 130, 138
- Tivoli Performance Viewer 118, 120, 133
 - Monitor garbage collections 120
- Tivoli Performance Viewer (TPV) 120
- TM RA
 - Common Client Interface 252
- TM Resource Adapter 14, 18, 98, 146, 170, 181, 229, 250, 263
- TM Resource Adapter (TMRA) 96, 185, 229
- TMEMBER 93, 101, 147, 159, 163, 173, 213, 215, 234
- TMEMBER name 211
- TMRA 14, 18, 36, 87, 114, 151, 157, 179, 226, 252, 257
- To-Be Model 42
- TPIPE 93–94, 108, 138, 147, 156, 159, 175, 182, 204, 228, 235, 237
- Tpipe 165, 199
- tpipe 148, 155, 199, 234–235, 237
- TPIPE function 218
- TPIPE name 93, 165, 211, 238
 - user ID 95
- tpipe name 211, 236–238
- transaction code 176, 205, 210, 253, 350
- Transaction Manager (TM) 12, 16, 118, 225
- Transaction Reporting Facility
 - increased precision 108
- Transaction Reporting Facility (TRF) 108
- transaction-pipe structure 161
- Transport Layer Security (TLS) 97
- TSO 79, 102, 111, 180
- tuning 15, 196
- two-phase commit 25, 198, 226, 298, 330, 332
- type-1 command 161

U

- UK23332 148
- UK23339 148
- UK25003 148
- UK25428 148
- UK27777 148
- UML 37, 106
- Unicode 290, 313, 350
- Universal Description, Discovery, and Integration (UDDI) 343, 345
- Universal Driver 321, 330–331
- UNIX System Services 74, 111, 124, 303
- UPDATE 95, 286, 334
- URL 9, 22, 74, 107–108, 119, 126, 133, 224, 279, 291, 310, 343
- user exit 95, 165, 190–191, 350

- user exits 190–191
- user Id 94, 108, 127, 147, 204, 236, 258
- user message exit 180–181
- Userid 93, 222
- UserName property 99

V

- VIEWHWS 83, 94, 97
- virtualization xv, 343, 346
- VSAM 63, 189
- VTAM 160, 248, 273

W

- Web 2.0 16, 21–22, 271–272
- Web browser 138, 273
- Web Enablement xvi, 19–20, 245, 248
- Web Oriented Architecture (WOA) 272
- Web page 138, 251, 265
- Web Service 9–11, 28, 58, 81, 100, 155, 185, 204, 225, 262, 279, 307, 310, 343, 347
 - one-way operation 211
 - request-response operation 214
 - secure manner 222
 - WSDL file 205
- Web service 8, 28, 100, 118, 146, 176, 185, 205, 226, 262, 265, 279, 306, 345
 - request message 187
 - response message 188, 210
 - secure manner 222
- Web Services 7, 10, 133, 146, 155, 171, 183–184, 203–204, 208, 225, 248–249, 279, 343
 - artifact 222
 - Business Process Execution Language 7
 - config 307
 - Description Language 17, 187, 309, 345
 - Distributed Management 343
 - for Remote Portlets 11
 - Service 188
 - Test Explorer 310
- Web Services (WS) 17, 185, 187, 204, 209, 226, 262, 279
- WebSphere xv, 14, 41, 58, 73, 86–87, 110–111, 119, 146, 155, 176, 183, 185, 221, 225–226, 248, 274, 294, 297, 323–324, 330, 339, 341–342, 358
- WebSphere Application
 - Server 19, 98, 120, 130, 159, 186, 227, 249, 297
- WebSphere Application Server 297
- WebSphere Application Server 16, 18, 28, 61–62, 89, 95, 114, 118, 146, 155–156, 159, 170, 183, 185–186, 225–226, 248–249, 297, 325, 362
 - logging and tracing 114
- WebSphere Application Server – Express 62
- WebSphere Application Server Community Edition 62
- WebSphere Application Server for z/OS 63, 100, 298
- WebSphere Application Server Network Deployment 63
- WebSphere Business Model 65
- WebSphere Business Monitor 53–54, 130, 134
- WebSphere Developer for zSeries (WDZ) 176
- WebSphere Enterprise Service Bus (WESB) 184

- WebSphere Integration Developer (WID) 14, 65, 184, 226, 251
- WebSphere MQ 151, 161, 347
- WebSphere Process Server (WPS) 184, 226, 229
- WebSphere Security 89
- WebSphere security 86
- WebSphere Service Registry and Repository 65
- WebSphere Software
 - V7.0 58
 - Version 7 159
- WebSphere Studio 111
- WebSphere Transformation Extender (WTX) 226–227
- WebSphere® DataPower SOA appliances 339
- WID 21, 184
- widget 280
- wirespeed performance 342–343
- workload balancing 167
- WS-BPEL 7
- WSDL 9, 37, 66, 72–73, 187, 204, 262, 279, 309, 343, 345
- WSDL file 10, 81, 205
- WSRP 10

X

- XCF 92, 95, 160, 171, 174, 176, 229, 324
- XCF group 160–161, 174
- XCF member 99–100, 174
- XIBAREA 101, 173–174
- XMI descriptions 303
- XMI file 100, 252, 256, 303, 312
 - MFS TABLE 256
- XMI repository 100, 252
 - physical location 100
- XML 7–10, 28, 37–38, 72, 101, 146, 157, 160, 171–172, 176, 205, 207, 245–246, 279, 293, 295, 339, 342–343
- XML adapter 146, 165, 175, 204–205, 210, 284, 287, 290
- XML conversion 185
 - HWSSOAP1 adapter exit 185
- XML converter 22, 185, 187, 206–207, 283, 286
- XML data 11, 23, 175, 187, 251, 302, 311–312
 - query collections 11
 - structure 208
- XML Denial of Service (XDOS) 343
- XML document 10, 146, 311–312, 343
 - content 314
 - data 314
 - IMS database segments 312
 - root element 314
 - tree representation 317
- XML message 8, 11, 175, 187, 206, 208, 210, 247, 344, 347
 - format 208, 343
- XML schema 23, 37, 264, 302
 - XML document 312–313, 316
- XML storage 314–315
- XML Stylesheet Language (XSL) 346
- XML tag 208, 312, 314
- xml version 205, 288, 308
- xmlns

- IMS 205, 288
- XQuery 11, 146, 293, 299, 311
- XQuery/XPath Data Model (XDM) 317

Z

- z/OS xv, 14–15, 59, 73, 86–87, 89, 105, 108, 124, 160, 167, 208, 223, 226–227, 250, 290, 294, 297, 323, 327



Powering SOA Solutions with IMS

(0.5" spine)
0.475" x 0.873"
250 x 459 pages



Powering SOA Solutions with IMS



Introduce yourself to how SOA concepts apply to IMS

Identify SOA implementation steps

Understand the newest SOA enhancements from IMS

New application development tools and the IBM® service-oriented architecture capabilities for IMS™ can help your business improve the speed and agility of its development efforts. Both IMS and the IMS SOA Integration Suite support your on demand systems and your distributed IMS application environment.

Powering SOA Solutions with IMS provides background and explanations to clarify the choices and methodologies that are available to modernize your IMS applications and provide access to IMS data stores through non-traditional callers.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7662-00

ISBN 0738432385